

ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ  
МАШИНЫ И СИСТЕМЫ



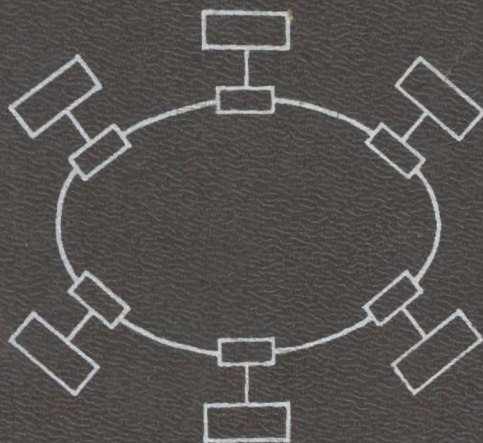
Б.М.Каган

# ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ И СИСТЕМЫ

---

Для студентов вузов

---



Б.М.Каган

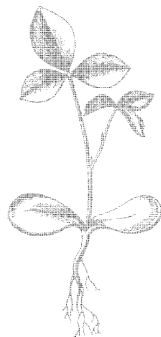
# ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ И СИСТЕМЫ

3-е издание, переработанное  
и дополненное

Допущено Государственным комитетом СССР по народному образованию в качестве учебного пособия для студентов вузов, обучающихся по специальностям: «Вычислительные машины, комплексы, системы и сети», «Автоматизированные системы обработки информации и управления» и «Программное обеспечение вычислительной техники и автоматизированных систем»



МОСКВА ЭНЕРГОАТОМИЗДАТ 1991



Scan AAW

ББК 32.973

К12

УДК 681.31 (075.8)

Рецензенты кафедры ЭВМ и ВС МГТУ им. Н. Э. Баумана

**Каган Б. М.**

**К12** Электронные вычислительные машины и системы: Учеб. пособие для вузов.— 3-е изд., перераб. и доп.— М.: Энергоатомиздат, 1991.— 592 с.: ил.

ISBN 5-283-01531-9

Третье издание книги (второе вышло в 1985 г.) переработано и дополнено материалами, отражающими новые идеи и архитектурные решения в современной вычислительной технике. Рассмотрены основы теории, принципы организации микропроцессоров, персональных компьютеров ЭВМ, ВС и вычислительных сетей.

Для студентов вузов и специалистов, занимающихся разработкой вычислительной техники и программных средств, их использованием для компьютеризации обработки информации и управления.

К  $\frac{2404090000-287}{051(01)-91}$  218-90

**ББК 32.973**

ISBN 5-283-01531-9

© Энергия, 1979  
© Энергоатомиздат, 1985, с изменениями  
© Автор, 1991, с изменениями



*Борис Моисеевич Каган, лауреат Государственной премии СССР, доктор технических наук, профессор — известный ученый в области электронной вычислительной техники и ее применения для автоматизации управления и в инженерном деле.*

*Автор внес большой вклад в создание оригинальных отечественных малых ЭВМ для инженерных расчетов и управления технологическими процессами.*

*Профессор Б. М. Каган является автором 130 научных трудов и изобретений, в том числе пользующихся широкой известностью монографий и учебных пособий (из них пять изданы за рубежом) по важным проблемам вычислительной техники.*

*В трудах проф. Б. М. Кагана исследованы и систематизированы вопросы архитектуры ЭВМ и вычислительных систем, в том числе управляющих вычислительных комплексов реального времени, отказоустойчивых систем, методы построения микропроцессорных устройств и систем автоматизации, запоминающих устройств большой емкости, организации систем связи ЭВМ с объектом управления в АСУ ТП, разработаны научные основы эксплуатации ЭВМ.*



## ПРЕДИСЛОВИЕ

Развитие электронной вычислительной техники, информатики и применение их средств и методов в народном хозяйстве, научных исследованиях, образовании и других сферах человеческой деятельности являются в настоящее время приоритетным направлением научно-технического прогресса. Это приводит к необходимости широкой подготовки специалистов по электронным вычислительным машинам, системам и сетям, программному обеспечению и прикладной математике, автоматизированным системам обработки данных и управления и другим направлениям, связанным с интенсивным использованием вычислительной техники. Всем этим специалистам необходимы достаточно глубокие знания принципов построения и функционирования современных электронных вычислительных машин, комплексов, систем и сетей, микропроцессорных средств, персональных компьютеров. Такие знания необходимы не только специалистам различных областей вычислительной техники, но и лицам, связанным с созданием программного обеспечения и применением ЭВМ в различных областях, что определяется тесным взаимодействием аппаратурных и программных средств в ЭВМ, тенденцией аппаратурной (в том числе микропрограммной) реализации системных и специализированных программных продуктов, позволяющей достигнуть увеличения производительности, надежности, функциональной гибкости, большей приспособленности вычислительных машин и систем к эксплуатационному обслуживанию.

В предыдущие десять лет одним из основных руководств при изучении в вузах студентами специальностей «ЭВМ», «АСУ», «Прикладная математика», вопросов архитектуры и принципов организации ЭВМ и систем были книги автора, изданные в 1979 г. (первое издание) и 1985 г. (второе издание).

Масштабы подготовки в вузах специалистов по ЭВМ, автоматизированным системам управления, программному обеспечению, прикладной математике и появление за последние годы многих новых важных идей и технических решений в электронной вычислительной технике, расширяющих области применения ЭВМ и способствующих вовлечению в активную работу по использованию ЭВМ, микропроцессорных средств и персональных компьютеров для управления процессами и обработки данных широкого круга инженеров, часто не имеющих специальной подготовки, делают целесообразным выпуск третьего, переработанного и дополненного издания книги «Электронные вычислительные машины и системы».

При переработке книги автор исходил из следующих соображений:

В последние годы «мир» электронных вычислительных машин значительно расширился — в нем наряду с машинами общего назначения заняли большое место супер-ЭВМ, малые ЭВМ и особенно микропроцессоры и микро-ЭВМ, персональные компьютеры. Чтобы не превращать курс по вычислительной технике в описание логической организации машин различных классов с неизбежным повторением одних и тех же вопросов, в книге обобщаются материалы, относящиеся к архитектурам вычислительных машин и микропроцессоров разных типов.

В новом издании книги внесены значительные изменения в структуру и содержание книги:

расширены разделы по элементам архитектуры процессоров, добавлен материал по новому стандарту для представления чисел с плавающей точкой, теговой организации памяти, использованию дескрипторов, по RISC-архитектуре (архитектуре с сокращенной системой команд), конвейерной обработке;

в книгу включена новая глава по архитектуре 8-, 16-, 32-разрядных микропроцессоров и персональных компьютеров;

включен новый раздел по принципам построения периферийных устройств персональных компьютеров;

включены глава по общим вопросам архитектуры ЭВМ общего назначения и раздел, в котором рассматриваются направления развития архитектуры в машинах ЕС ЭВМ и новых зарубежных разработках в этой области;

расширен материал по организации многомашинных ВС и микропроцессорных суперЭВМ, в том числе векторно-конвейерных и с управлением потоком данных;

в главу по организации интерфейсов включены материалы по интерфейсам «Q-bus» и «Multibus-II»; полностью переработана и значительно расширена глава по вычислительным сетям.

Некоторые разделы предыдущего издания подверглись сокращениям.

Автор выражает глубокую благодарность д-ру техн. наук, проф. Ю. М. Смирнову, доц. Л. В. Суркову, всему коллективу кафедры ЭВМ и ВС МГТУ им. Н. Э. Баумана, а также редактору книги доц. В. Г. Першееву за ценные советы и замечания, которые способствовали улучшению содержания данного издания книги.

Автор будет весьма признателен всем читателям, приславшим свои замечания и пожелания по адресу: 113114, Москва, М-114, Шлюзовая наб., 10, Энергоатомиздат.

*Автор*

## ОСНОВНЫЕ ПОНЯТИЯ

### 1.1. Два класса ЭВМ

Любая форма человеческой деятельности, любой процесс функционирования технического объекта связаны с передачей и преобразованием информации.

Одно из важнейших положений кибернетики состоит в том, что без информации и ее переработки невозможны организованные системы, какими являются живые организмы и искусственные, созданные человеком технические системы.

*Информацией* называют сведения о тех или иных явлениях природы, событиях в общественной жизни и процессах в технических устройствах. Информация, воплощенная и зафиксированная в некоторой материальной форме, называется *сообщением*. Например, при планировании и управлении производством собираются и обрабатываются сведения (сообщения) о потребностях в той или иной продукции, ресурсах рабочей силы, сырья и материалов, производительности станков и другого оборудования и вырабатываются соответствующие управляющие решения.

Сообщения могут быть непрерывными и дискретными (цифровыми).

*Непрерывное (аналоговое) сообщение* представляется некоторой физической величиной (электрическим напряжением, током и др.), изменения которой во времени отображают протекание рассматриваемого процесса, например изменения температуры в нагревательной печи. Физическая величина, передающая непрерывное сообщение, может в определенном интервале принимать любые значения и изменяться в произвольные моменты времени.

Для *дискретных сообщений* характерно наличие фиксированного набора элементов, из которых в некоторые моменты времени формируются различные последовательности. Важным является не физическая природа элементов, а то обстоятельство,

что набор элементов конечен и поэтому любое дискретное сообщение конечной длины передает конечное число значений некоторой величины.

Элементы, из которых состоит дискретное сообщение, называют *буквами* или *символами*. Набор этих букв образует *алфавит*. Здесь под буквами в отличие от обычного представления понимаются любые знаки (обычные буквы, цифры, знаки препинания, математические и прочие знаки), используемые для представления дискретных сообщений.

При дискретной форме представления информации отдельным элементам ее могут быть присвоены числовые (цифровые) значения. В таких случаях говорят о *цифровой (числовой) информации*.

Передача и преобразования дискретной информации любой формы (например, обычного текста, содержащего обычные буквы и цифры) могут быть сведены к эквивалентным передаче и преобразованиям цифровой информации. Более того, возможно с любой необходимой степенью точности непрерывные сообщения заменять цифровыми путем квантования непрерывного сообщения по уровню и времени. Таким образом, любое сообщение может быть представлено в цифровой форме.

*Электронные вычислительные машины* или *компьютеры* являются преобразователями информации. В них исходные данные задачи преобразуются в результат ее решения. В соответствии с используемой формой представления информации машины делятся на два класса: *непрерывного действия — аналоговые и дискретного действия — цифровые*.

В силу универсальности цифровой формы представления информации цифровые электронные вычислительные машины представляют собой наиболее универсальный тип устройства обработки информации. Именно эти машины, в дальнейшем называемые сокращенно ЭВМ, составляют предмет изучения в настоящей книге.

Замечательные свойства ЭВМ — автоматизация вычислительного процесса на основе программного управления, огромная скорость выполнения арифметических и логических операций, возможность хранения большого количества различных данных, возможность решения широкого круга математических задач и задач обработки данных — делают эти машины мощным средством научно-технического прогресса.

Особое значение ЭВМ состоит в том, что впервые с их появлением человек получил орудие для автоматизации процессов обработки информации. Во многих случаях ЭВМ позволяют существенно повысить эффективность умственного труда.

Внедрение ЭВМ оказало большое влияние на многие области науки и техники, вызвало процесс их математизации и компьютеризации.

## 1.2. Принципы действия ЭВМ

Упрощенная структура ЭВМ представлена на рис. 1.1. ЭВМ содержит следующие основные устройства: арифметическо-логическое устройство, память, управляющее устройство, устройство ввода данных в машину, устройство вывода из нее результатов расчета и пульт ручного управления.

*Арифметическо-логическое устройство (АЛУ)* производит арифметические и логические преобразования над поступающими в него *машинными словами*, т. е. кодами определенной длины, представляющими собой числа или другой вид информации. Количество разрядов в машинном слове обычно совпадает с количеством разрядов в основных регистрах АЛУ.

*Память* хранит информацию, передаваемую из других устройств, в том числе поступающую в машину извне через устройство ввода, и выдает во все другие устройства информацию, необходимую для протекания вычислительного процесса. Память машины в большинстве случаев состоит из двух существенно отличающихся по своим характеристикам частей: быстродействующей *основной* или *оперативной* (внутренней) *памяти* (ОП) и сравнительно медленно действующей, но способной хранить значительно больший объем информации *внешней памяти* (ВнП).

Оперативная память содержит некоторое число ячеек, каждая из которых служит для хранения машинного слова или его части. Ячейки нумеруются, номер ячейки называется *адресом*.

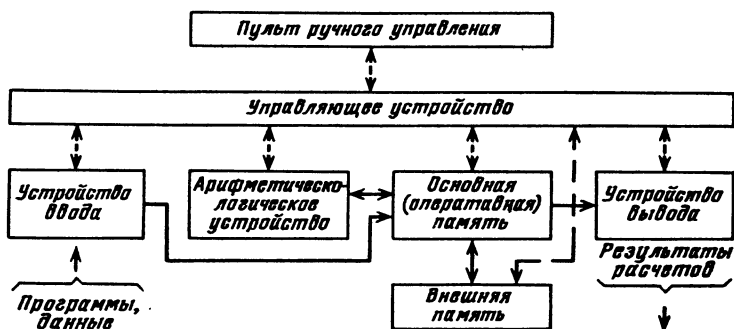


Рис. 1.1. Структура электронной цифровой вычислительной машины



В *запоминающих устройствах*, реализующих в ЭВМ функцию памяти, выполняются операции считывания хранимой информации для передачи в другие устройства и записи информации, поступающей из других устройств. При считывании слова из ячейки содержимое последней не меняется и при необходимости слово может быть снова взято из той же ячейки. При записи хранившееся в ячейке слово стирается и его место занимает новое.

Непосредственно в вычислительном процессе участвует только ОП, и лишь после окончания отдельных этапов вычислений из ВнП в ОП передается информация, необходимая для следующего этапа решения задачи.

*Управляющее устройство (УУ)* автоматически без участия человека управляет вычислительным процессом, посылая всем другим устройствам сигналы, предписывающие им те или иные действия, в частности включает АЛУ на выполнение нужной операции.

*Алгоритм* решения задачи численным методом называют последовательность арифметических и логических операций, которые надо произвести над исходными данными и промежуточными результатами для получения решения задачи. Поэтому алгоритм можно задать указанием, какие следует произвести операции, в каком порядке и над какими словами. Описание алгоритма в форме, воспринимаемой ЭВМ, называется *программой*.

Программа состоит из отдельных *команд*. Каждая команда предписывает определенное действие и указывает, над какими словами (операндами) это действие производится. Программа представляет собой совокупность команд, записанных в определенной последовательности, обеспечивающей решение задачи на ЭВМ.

Пусть, например, нужно вычислить

$$F = (a - x) / (ax + c)$$

при заданных численных значениях  $a$ ,  $c$ ,  $x$ . Программу вычислений  $F$  можно представить следующей последовательностью команд:

- 1) вычесть из числа  $a$  число  $x$ ;
- 2) умножить число  $a$  на число  $x$ ;
- 3) прибавить к результату действия второй команды число  $c$ ;
- 4) разделить результаты действия первой команды на результат действия третьей команды.

Чтобы устройство управления могло воспринять команды, они должны быть закодированы в цифровой форме.

Автоматическое управление процессом решения задачи достигается на основе *принципа программного управления*, являющегося основной особенностью ЭВМ.

Другим важнейшим принципом является *принцип хранимой в памяти программы*. Согласно этому принципу команды программы, закодированные в цифровом виде, хранятся в памяти наравне с числами. В команде указываются не сами участвующие в операциях числа, а адреса ячеек ОП, в которых они находятся, и адрес ячейки, куда помещается результат операции.

Поскольку программа хранится в памяти, одни и те же команды могут нужное число раз извлекаться из памяти и выполняться. Более того, так как команды представляются в машине в форме чисел, то над командами как над числами машина может производить операции («модификации команд»).

Использование электронных схем, принципов программного управления и хранимой в памяти программы позволило достигнуть высокого быстродействия и сократить во много раз число команд в программах решения задач, содержащих вычислительные циклы, по сравнению с числом операций, которые производит машина при выполнении этих программ.

Команды выполняются в порядке, соответствующем их расположению в последовательных ячейках памяти, кроме команд безусловного и условного переходов, изменяющих этот порядок соответственно безусловно или только при выполнении некоторого условия, обычно задаваемого в виде равенства нулю, положительного или отрицательного результата предыдущей команды или отношения типа  $>$ ,  $=$ ,  $<$  для указываемых командой чисел. Именно благодаря наличию команд условного перехода ЭВМ может автоматически изменять соответствующим образом ход вычислительного процесса, решать сложные логические задачи.

Перед решением задачи на ЭВМ программа и исходные данные должны быть помещены в ее память. Предварительно эта информация наносится на перфоленту, перфокарты в виде соответствующих комбинаций отверстий на них или на магнитную ленту путем соответствующего намагничивания участков ее поверхности. Затем при помощи *устройства ввода* программа и исходные данные считываются с перфокарт, перфоленты или магнитной ленты и переносятся в ОП. Информация, необходимая для решения задачи, может вводиться в ОП непосредственно с клавиатуры дисплея или электрифицированной пишущей машины.

*Устройство вывода* служит для выдачи из машины результатов расчета, например, путем печатания их на электрифицированных печатных устройствах или отображения на экране дисплея.

При помощи *пульта управления* оператор пускает и останавливает машину, а при необходимости может вмешиваться в процесс решения задачи.

ЭВМ обладают универсальностью, они пригодны для решения разнообразных задач. Любая необходимая точность вычислений может быть достигнута путем увеличения числа разрядов в представлении чисел в ЭВМ и соответствующего увеличения количества электронного оборудования без повышения требований к точности работы самих электронных схем.

Представленная на рис. 1.1 структура (модель) вычислительной машины, получившая название фоннеймановской \*, благодаря ее изящной простоте и большой гибкости при управлении вычислительным процессом с самых первых шагов электронной вычислительной техники и по сей день доминирует при построении различных ЭВМ.

Однако в последние годы конструкторы ЭВМ, стремясь достигнуть существенного повышения их производительности, в ряде случаев отходят от модели фон Неймана (см. гл. 15).

Приведем один из примеров. В фоннеймановской машине с общей памятью для данных и команд имеется всего одна шина (магистраль) для передачи из памяти в другие устройства команд и данных, что ведет к снижению скорости работы ЭВМ.

Возможно построение машины с отдельными памятьями и шинами для хранения и передачи команд и данных, допускающей параллельное во времени извлечение их из памяти и передачу по шинам. Такая структура (модель) получила название гарвардской, так как была реализована впервые в 1944 г. в Гарвардском университете (США) в ранней релейной вычислительной машине, предшествовавшей появлению электронных вычислительных машин. Гарвардская модель реализована, в частности, в некоторых микропроцессорах.

### **1.3. Понятие о системе программного (математического) обеспечения ЭВМ. Понятие об архитектуре ЭВМ**

Для придания ЭВМ определенных свойств используют средства двух видов: аппаратные и программные. Последние называются также *средствами программного (математического) обеспечения* <sup>1</sup>.

\* Принцип построения ЭВМ с программой, хранимой в общей оперативной памяти вместе с данными, впервые предложен в 1945 г. американским математиком Дж. фон Нейманом.

<sup>1</sup> В зарубежной литературе терминам «аппаратные средства» и «средства математического обеспечения» соответствуют термины *hardware* и *software*, что буквально означает «жесткий товар» и «мягкий товар».

Часть свойств ЭВМ приобретает благодаря наличию в ее составе электронного или электромеханического оборудования, специально предназначенного для реализации этих свойств. Арифметическо-логическое устройство машины является примером аппаратурных средств.

Ряд других свойств реализуется без специальных аппаратурных средств программным путем, при этом используются имеющиеся аппаратурные средства машины, работающие в предписанном порядке в соответствии с программой, обеспечивающей выполнение машиной данной функции. Машина может не иметь, например, аппаратурно реализованной операции извлечения квадратного корня. Однако если имеется программа извлечения квадратного корня, использующая наличные аппаратурные средства машины, то с точки зрения пользователя машина обладает свойством извлекать квадратный корень.

При помощи аппаратурных средств соответствующие функции выполняются значительно быстрее, чем программным путем. Поэтому одна и та же функция может реализоваться в малых моделях ЭВМ при помощи программных средств, а в больших, для которых быстродействие является одной из важнейших характеристик,— аппаратурными средствами.

Часто для придания ЭВМ того или иного свойства используют комбинацию аппаратурных и программных средств, что позволяет при сравнительно небольших аппаратурных затратах достигнуть высокой эффективности и быстродействия при выполнении соответствующей функции.

Средства программного обеспечения и аппаратурные средства являются двумя взаимосвязанными компонентами современной вычислительной техники.

*Система программного (математического) обеспечения ЭВМ* представляет собой комплекс программных средств, в котором можно выделить операционную систему, комплект программ технического обслуживания и пакеты прикладных программ (рис. 1.2).

*Операционные системы* являются важнейшей и центральной частью программного обеспечения ЭВМ, предназначенной для эффективного управления вычислительным процессом, планирования работы и распределения ресурсов ЭВМ, автоматизации процесса подготовки программ и организации их выполнения при различных режимах работы машины, облегчения общения оператора с машиной.

Пользователи и операторы не имеют прямого доступа к устройствам ЭВМ. Связь пользователей и операторов с ЭВМ (точнее, с ее аппаратурными средствами) производится при

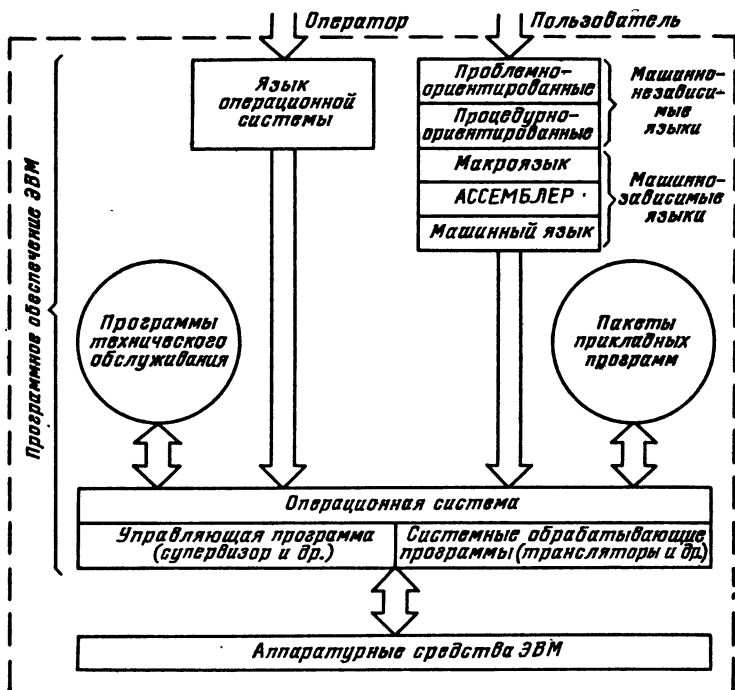


Рис. 1.2. Вычислительная система — совокупность аппаратурных и программных средств ЭВМ

помощи операционной системы, обеспечивающей определенный уровень общения человека с машиной.

Уровень общения в первую очередь определяется уровнем языка, на котором оно происходит. Современные операционные системы содержат трансляторы с языков различного уровня, таких, как АССЕМБЛЕР, ФОРТРАН, ПЛ-1, ПАСКАЛЬ, Си и др.

Комплект *программ технического обслуживания*, предназначенный для уменьшения трудоемкости эксплуатации ЭВМ, содержит программы проверки работоспособности ЭВМ и отдельных ее устройств, определения (диагностирования) мест неисправностей.

*Пакеты прикладных программ* (ППП) представляют собой структурированные комплексы программы (часто со специализированными языковыми средствами), предназначенные для решения определенных достаточно широких классов задач (научно-технических, планово-экономических и др.), а также для расширения функций операционных систем (управление базами

данных, реализация режимов телеобработки данных, реального времени и др.).

Аппаратурные средства ЭВМ и система ее программного обеспечения в совокупности образуют *вычислительную систему*.

При создании новой ЭВМ разработка аппаратуры и программного обеспечения должна производиться одновременно и взаимосвязанно.

В настоящее время средства программного обеспечения должны рассматриваться как вид промышленной продукции отрасли вычислительного машиностроения — такой же, как и сами ЭВМ. В соответствии с этим должны применяться промышленные методы при разработке, испытаниях, проверке, размножении средств программного обеспечения. Вычислительные машины должны поставляться комплектно с системами программного обеспечения, представленными в такой форме и на таких носителях информации, чтобы использование этих средств не вызывало затруднений.

Сложность современных вычислительных машин закономерно привела к понятию *архитектуры вычислительной машины*, охватывающей комплекс общих вопросов ее построения, существенных в первую очередь для пользователя, интересующегося главным образом возможностями машины, а не деталями ее технического исполнения.

Круг вопросов, подлежащих решению при разработке архитектуры ЭВМ, можно условно разделить на вопросы общей структуры, организации вычислительного процесса и общения пользователя с машиной, вопросы логической организации представления, хранения и преобразования информации и вопросы логической организации совместной работы различных устройств, а также аппаратурных и программных средств машины.

## 1.4. Поколения ЭВМ

В середине нашего века развитие атомной физики, ракетной и космической техники потребовало решения вычислительных задач такого большого объема, что с ними нельзя было справиться при помощи имевшихся в то время клавишных или перфорационных счетных машин. Эта потребность привела к созданию на рубеже 40—50-х годов цифровых электронных вычислительных машин.

Идея использования программного управления для построения устройства, автоматически выполняющего арифметические вычисления, была впервые высказана английским математиком Ч. Бэббиджем еще в 1833 г. Однако его попытки построить



механическое вычислительное устройство с программным управлением не увенчались успехом.

Фактически эта идея была реализована лишь спустя 100 лет, когда в 1942 г. К. Цюзе в Германии и в 1944 г. Г. Айкен в США построили на электромагнитных реле вычислительные машины с управлением от перфоленты [83].

Идея программного управления вычислительным процессом была существенно развита американским математиком Дж. фон Нейманом, который в 1945 г. сформулировал принцип построения вычислительной машины с хранимой в памяти программой.

Первые ЭВМ с программным управлением и с хранимой в памяти программой появились практически одновременно в Англии, США и СССР.

Фундаментальный вклад в развитие отечественной вычислительной техники внес акад. С. А. Лебедев. Под его руководством в 1949—1951 гг. в АН УССР в Киеве была построена первая в нашей стране ЭВМ — Малая Электронная Счетная Машина (МЭСМ), а в 1952—1954 гг. в ИТМ и ВТ АН СССР — Быстродействующая Электронная Счетная Машина (БЭСМ), выполнявшая 8000 операций/с и являвшаяся в то время одной из самых быстродействующих ЭВМ в мире. Последующие годы акад. С. А. Лебедев посвятил созданию оригинальных советских ЭВМ высокой производительности.

Одну из первых в стране ЭВМ построили в начале 50-х годов чл.-кор. АН СССР И. С. Брук и его сотрудники Н. Я. Матюхин и М. А. Карцев в Энергетическом институте АН СССР в Москве. Первая выпускавшаяся промышленностью ЭВМ «Стрела» была разработана научным коллективом, руководимым Ю. Я. Базилевским.

Советские ученые, в первую очередь академики С. А. Лебедев, М. В. Келдыш, В. М. Глушков, В. С. Семенихин, и их научные школы внесли крупный вклад в развитие принципов построения и теории ЭВМ и их программного обеспечения, методов использования ЭВМ в народном хозяйстве.

На протяжении четырех десятилетий электронная вычислительная техника бурно развивается. На наших глазах появились, сменяя друг друга, несколько поколений ЭВМ. Появление новых поколений ЭВМ вызывалось расширением областей и развитием методов их применения, требовавших более производительных, более дешевых и более надежных машин.

*Поколение ЭВМ* определяется совокупностью взаимосвязанных и взаимообусловленных существенных особенностей и характеристик используемой при построении машин конструктивно-технологической (в первую очередь элементной) базы и реа-

лизуемой в машине архитектуры (логической организации).

Первое поколение образовали ламповые ЭВМ, промышленный выпуск которых начался в начале 50-х годов. В качестве компонентов логических элементов использовались электронные лампы.

К первому поколению ЭВМ относятся созданные советскими учеными и инженерами ламповые вычислительные машины БЭСМ-2, «Стрела», М-2, М-3, «Минск-1», «Урал-1», «Урал-2», М-20, в основном ориентированные на решение научно-технических задач.

Ламповые ЭВМ потребляли большую мощность, имели большие габаритные размеры, малую емкость оперативной памяти и, что особенно важно, невысокую надежность, в первую очередь из-за частого выхода из строя электронных ламп.

В вычислительных машинах второго поколения, появившихся в конце 50-х годов, полупроводниковые приборы — транзисторы — заменили электронные лампы, что существенно повысило надежность, снизило потребление мощности, уменьшило габаритные размеры ЭВМ. Это позволило создать ЭВМ, обладающие большими логическими возможностями и более высокой производительностью. Наряду с машинами для научных расчетов появились ЭВМ для решения планово-экономических задач (задач обработки данных) и управления производственными процессами.

В нашей стране были созданы полупроводниковые ЭВМ различного назначения: малые ЭВМ серий «Наири» и «Мир», средние ЭВМ для научных расчетов и обработки данных со скоростью работы 5—30 тыс. операций/с — «Минск-2», «Минск-22», «Минск-32», «Урал-14», «Раздан-3», БЭСМ-4, М-220 — и управляющие вычислительные машины «Днепр», ВНИИЭМ-3 и др.

В рамках второго поколения академики С. А. Лебедев и В. А. Мельников создали сверхбыстродействующую ЭВМ БЭСМ-6 с производительностью 1 млн. операций/с.

Второе поколение ЭВМ позволило существенно расширить сферу использования вычислительной техники, приступить к созданию АСУ отраслями, предприятиями и технологическими процессами.

Стремление к повышению надежности, быстродействия, снижению стоимости аппаратуры привело к появлению новой элементной базы вычислительной техники в виде интегральных микросхем, на основе которых были созданы ЭВМ третьего поколения.

ЭВМ третьего поколения появились во второй половине 60-х годов, когда фирма IBM (США) разработала систему машин IBM-360. Эта система оказала влияние на логическую

организацию машин общего назначения третьего поколения, разработанных в других странах.

Советский Союз и другие страны — члены СЭВ в начале 70-х годов совместно разработали и организовали серийное производство Единой Системы ЭВМ (ЕС ЭВМ) и Системы Малых ЭВМ (СМ ЭВМ) — машин третьего поколения на интегральных микросхемах.

В машинах третьего поколения значительное внимание уделено уменьшению трудоемкости подготовки программ для решения задач на ЭВМ, облегчению связи оператора с машиной, повышению эффективности использования дорогостоящего оборудования машин, облегчению эксплуатационного обслуживания ЭВМ, что достигается при помощи соответствующих операционных систем.

В последние годы появились ЭВМ и вычислительные устройства, которые следует отнести к четвертому поколению. Контуры этого поколения довольно трудно четко определить, так как в настоящее время оно представлено главным образом новыми, ранее не существовавшими типами вычислительных средств и лишь в отношении некоторых вопросов (например, замена ферритовых памятей полупроводниковыми) затронуло машины общего назначения, выполняющие основной объем вычислительных работ в разного рода вычислительных центрах.

Конструктивно-технологической основой ЭВМ четвертого поколения являются интегральные микросхемы с большей (БИС) и сверхбольшей (СБИС) степенями интеграции, содержащие тысячи, десятки и сотни тысяч транзисторов на одном кристалле. В первую очередь на БИС строят памяти ЭВМ.

К четвертому поколению относятся реализованные на СБИС такие новые средства вычислительной техники, как микропроцессоры и создаваемые на их основе микроЭВМ. Микропроцессоры и микроЭВМ нашли широкое применение в устройствах и системах автоматизации измерений, обработки данных и управления технологическими процессами, при построении различных специализированных цифровых устройств и машин.

Вычислительные возможности микроЭВМ оказались достаточно точными для создания на их основе в рамках ЭВМ четвертого поколения, нового по ряду эксплуатационных характеристик и способу использования типа вычислительных устройств, — *персональных ЭВМ (персональных компьютеров)*, получивших в настоящее время широкое распространение.

В ЭВМ четвертого поколения достигается дальнейшее упрощение контактов человека с ЭВМ путем повышения уровня машинного языка, значительного расширения благодаря применению микропроцессоров функций устройств (терминалов),

используемых человеком для связи с ЭВМ, начинается практическая реализация голосовой связи с ЭВМ. Использование БИС позволяет аппаратурными средствами реализовать некоторые функции программ операционных систем (аппаратурная реализация трансляторов с алгоритмических языков высокого уровня и др.), что способствует увеличению производительности машин.

Характерным для крупных ЭВМ четвертого поколения является наличие нескольких процессоров, ориентированных на выполнение определенных операций, процедур или на решение некоторых классов задач. В рамках этого поколения создаются многопроцессорные вычислительные системы с быстродействием в несколько сотен миллионов и более операций в секунду и многопроцессорные управляющие комплексы повышенной надежности (см. гл. 15).

Примерами крупных вычислительных систем, которые следует отнести к четвертому поколению, являются многопроцессорные комплексы «Эльбрус-2» и «Эльбрус-3» с суммарным быстродействием соответственно 100 млн. и порядка 1 млрд. операций/с и многопроцессорная вычислительная система ПС-2000, содержащая до 64 процессоров, управляемых общим потоком команд, в которой при распараллеливании процесса выполнения программ может быть достигнуто быстродействие до 200 млн. операций/с.

В последнее время определились контуры нового, пятого поколения ЭВМ. В значительной степени этому способствовали публикации сведений о проекте ЭВМ пятого поколения, разрабатываемом ведущими японскими фирмами и научными организациями.

Согласно этому проекту ЭВМ и вычислительные системы пятого поколения помимо более высокой производительности и надежности при более низкой стоимости должны обладать следующими качественно новыми свойствами: возможностью взаимодействия с ЭВМ при помощи естественного языка, человеческой речи и графических изображений; способностью системы обучаться, производить ассоциативную обработку информации, делать логические суждения, вести «разумную» беседу с человеком в форме вопросов и ответов; способностью системы «понимать» содержимое базы данных, которая при этом превращается в «базу знаний», и использовать эти «знания» при решении задач.

Предполагается, что в ЭВМ пятого поколения быстродействие машин и емкость основной памяти составят соответственно 2 млн. операций/с и 0,5—5 Мбайт для персональных компьютеров и 1—100 млрд. операций/с и до 160 Мбайт для сверхпроизводительных ЭВМ.

Ожидается, что в машинах пятого поколения будут использоваться интегральные микросхемы со сверхбольшой степенью интеграции, содержащие до 1—10 млн. транзисторов на кристалле. Для разработки таких схем потребуются мощные системы автоматизации проектирования.

## 1.5. Основные характеристики ЭВМ

Важнейшими эксплуатационными характеристиками ЭВМ являются ее *производительность  $P$  и общий коэффициент эффективности машины*:

$$\mathcal{E} = P / (C_{\text{ЭВМ}} + C_{\text{экс}}), \quad (1.1)$$

представляющий собой отношение ее производительности к сумме стоимости самой машины  $C_{\text{ЭВМ}}$  и затрат на ее эксплуатацию за определенный период времени (например, период окупаемости капитальных затрат)  $C_{\text{экс}}$ .

Так как часто бывает трудно оценить затраты на эксплуатацию данной модели ЭВМ, то оценивают эффективность машины по упрощенной формуле

$$\mathcal{E}' = P / C_{\text{ЭВМ}}. \quad (1.2)$$

Оценка и сопоставление производительности различных ЭВМ представляют собой сложную проблему, в сущности не получившую до сих пор удовлетворительного решения.

Под производительностью ЭВМ можно было бы понимать количество «вычислительной работы», или, другими словами, количество задач, выполняемых машиной в единицу времени. Однако на производительность ЭВМ, оцениваемую по числу решенных в единицу времени задач, влияет слишком много факторов, в том числе тип задач, стиль программирования и другие особенности программ, логические возможности системы команд, особенности операционной системы, структура процессора, характеристики и организация оперативной и внешней памяти, системы ввода-вывода, состав и характеристики периферийных устройств и др.

В этих условиях оценка и сопоставление производительности машин по некоторому набору задач дают сравнительно достоверный результат, если машина предназначена для решения определенного круга задач. Примером такого подхода служит сопоставление производительности различных машин на основе использования стандартной программы решения систем линейных уравнений LINPACK в среде ФОРТРАНА. Применяются для этой цели и некоторые другие пакеты программ.

Однако большей частью производительность ЭВМ общего назначения оценивается упрощенно по скорости выполнения некоторых «смесей» команд, формируемых путем анализа частоты исполнения разных команд при выполнении программ решения задач некоторого класса, например научно-технических или планово-экономических задач. На основе такого анализа отдельным видам команд присваиваются определенные весовые коэффициенты.

При использовании смесей задач производительность ЭВМ (число операций в секунду) определяется по формуле

$$P = \sum_{s=1}^h k_s / \sum_{s=1}^h k_s t_s, \quad (1.3)$$

где  $k_s$  и  $t_s$  — соответственно весовой коэффициент и продолжительность выполнения  $s$ -й команды;  $h$  — число различных команд в «смеси».

Используют «смесь Гибсона» для оценки производительности при решении научно-технических задач и «смесь GRO-WO» для планово-экономических и информационно-логических задач.

ГОСТ 16325—88 предусматривает определение производительности ЭВМ общего назначения по «смеси алгоритмических действий» по выражению, аналогичному (1.3). В ГОСТ 16325—88 приведены стандартные смеси — наборы алгоритмических действий и их весовых коэффициентов для научно-технических задач и планово-экономических расчетов. Эти смеси могут использоваться для оценки производительности ЭВМ, имеющих уровень системы команд не выше уровня приводимых в стандарте алгоритмических действий.

Использование смесей алгоритмических действий вместо смесей команд способствует сопоставимости расчетных значений производительности ЭВМ с различными системами команд.

Наряду со сказанным выше часто используют и более примитивную оценку производительности (быстродействия) ЭВМ — количество коротких операций (типа регистр — регистр) в секунду, которая лишь в малой степени характеризует действительную производительность машины, скорее, косвенно оценивает ее тактовую частоту.

Производительность высокопроизводительных ЭВМ, главным образом ориентированных на научно-технические задачи, связанные с выполнением больших объемов вычислений с много-разрядными (32, 64 и даже 128 разрядов) числами с плавающей точкой, часто оценивают в мегафлопах (Мфлоп/с — миллион



операций с плавающей точкой в секунду)<sup>1</sup>. Можно отметить, что и эта характеристика не дает исчерпывающего представления о возможности машины, если принять во внимание, что на высокопроизводительные ЭВМ все в большей степени возлагают функции управления большими базами данных, связанные с большим количеством информационно-логических операций и операций обмена данными с внешней памятью.

К более частным характеристикам ЭВМ относятся:

число разрядов в машинном слове (влияет на точность вычислений и диапазон представимых в машине чисел);

скорость выполнения основных видов команд;

емкость оперативной памяти;

максимальная скорость передачи информации между ядром ЭВМ (процессор и оперативная память) и внешним (периферийным) оборудованием.

Однако все приведенные выше характеристики представляют собой лишь номинальные показатели. Они реализуются, если ЭВМ полностью работоспособна. Представление о реальных возможностях выполнения машиной определенных функций, решения определенного комплекса задач, особенно задач управления, может быть получено только с учетом эксплуатационной надежности машины. Эксплуатационная надежность непосредственно влияет и на стоимость эксплуатационного обслуживания ЭВМ.

Надежность ЭВМ определяется частотой нарушения ее работоспособности из-за отказов и сбоев, затратами времени на их устранение (подробнее см. в гл. 12).

Пусть  $K_n$  — комплексный коэффициент эксплуатационной надежности (комплексный коэффициент использования) машины, характеризующий потерю производительности из-за нарушения надежности функционирования ЭВМ. Тогда с учетом надежности машины формулы (1.1) и (1.2) примут вид

$$\mathcal{E} = PK_n / (C_{\text{ЭВМ}} + C_{\text{экс}}); \quad (1.4)$$

$$\mathcal{E}' = PK_n / C_{\text{ЭВМ}}. \quad (1.5)$$

Повышение надежности вычислительной машины, оцениваемое приращением  $\Delta K_n$ , связано с дополнительными затратами  $\Delta C_{\text{ЭВМ}}$ , эффективность которых можно оценить отношением

$$\mathcal{E}_n = \Delta K_n / \Delta C_{\text{ЭВМ}}. \quad (1.6)$$

---

<sup>1</sup> От английской аббревиатуры MFLOP/s — million of floating point operations per second.

При создании новых ЭВМ должно обеспечиваться значительное возрастание отношений производительность/стоимость и надежность/стоимость.

## 1.6. Основные области применения вычислительной техники и основные типы ЭВМ

Развитие вычислительной техники, сферы и методов ее использования — процессы взаимосвязанные и взаимообусловленные. С одной стороны, потребности народного хозяйства, науки и культуры стимулируют поиски учеными и конструкторами новых путей построения ЭВМ, а с другой стороны, появление электронных вычислительных машин, систем и устройств с большими функциональными возможностями, с существенно улучшенными показателями по производительности, стоимости, габаритным размерам, надежности и т. п. создает предпосылки для непрерывного расширения областей и развития форм применения ЭВМ.

Первоначально сравнительно узкая сфера применения ЭВМ, главным образом для научных и технических расчетов, в ко-

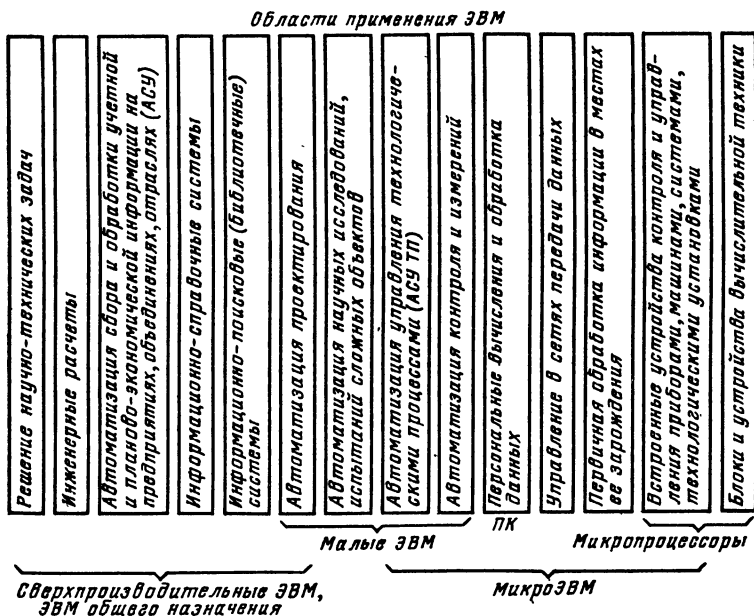


Рис. 1.3. Области применения и основные типы ЭВМ

роткий срок существенно расширилась и охватила почти все области науки, техники, планирования и управления производством, контроля и управления технологическими процессами, все области человеческой деятельности, связанные с обработкой больших объемов информации.

Разнообразие областей и форм использования ЭВМ породило широкий спектр требований к характеристикам и особенностям организации машин и систем. В результате к настоящему времени определились основные типы ЭВМ, которые при сохранении (в основном) указанных в § 1.2 фундаментальных принципов существенно разнятся не только по количественным характеристикам, но и по архитектуре, электронно-технологической базе и используемым периферийным устройствам.

Основные средства современной вычислительной техники можно классифицировать следующим образом: сверхпроизводительные ЭВМ и системы (суперЭВМ), ЭВМ общего назначения, малые ЭВМ, микроЭВМ, персональные компьютеры, микропроцессоры.

На рис. 1.3 типы машин (вычислительных средств) соотнесены с основными областями применения ЭВМ. Приведенное разделение весьма условно, границы между типами машин быстро меняются под влиянием успехов в области микроэлектроники и архитектуры ЭВМ, тем более что в ряде применений машины разных типов объединяются в вычислительные системы и комплексы различных конфигураций.

### **1.6.1. СуперЭВМ**

К сверхпроизводительным машинам (системам), т. е. к *супер-ЭВМ*, в настоящее время относят машины (системы) с производительностью в несколько сотен или тысяч мегафлоп в секунду (Мфлоп/с).

Подобные машины используются для решения особенно сложных научно-технических задач, задач обработки больших объемов данных в реальном масштабе времени, поиска оптимальных решений в задачах экономического планирования и автоматического проектирования сложных объектов.

В архитектурах суперЭВМ обнаруживается ряд принципиальных отличий от классической фоннеймановской модели ЭВМ. Принципы организации сверхпроизводительных вычислительных систем — суперЭВМ — рассмотрены в гл. 15.

### 1.6.2. ЭВМ общего назначения

Современные ЭВМ общего назначения имеют высокую, почти выравненную производительность при решении как научно-технических, так и информационно-логических задач (задач обработки данных).

Для научно-технических расчетов характерными являются относительно небольшие объемы входной (исходных данных) и выходной (результатов расчета) информации и очень большое количество вычислительных операций с плавающей точкой, выполняемых с высокой точностью над многоразрядными (32-, 64-, 128-разрядными двоичными) словами (числами).

Иной характер носят задачи обработки данных (планово-экономические, информационно-логические, учета, статистики), решение которых связано с необходимостью ввода в машину, запоминания, обработки очень большого количества данных — десятичной и текстовой алфавитно-цифровой информации, представляемой в машине словами переменной длины. Сама обработка состоит в выполнении над порциями данных сравнительно небольшого числа логических и арифметических операций. Результаты обработки должны печататься и отображаться в отредактированной форме в виде таблиц, ведомостей, графиков, алфавитно-цифровых текстов и т. п.

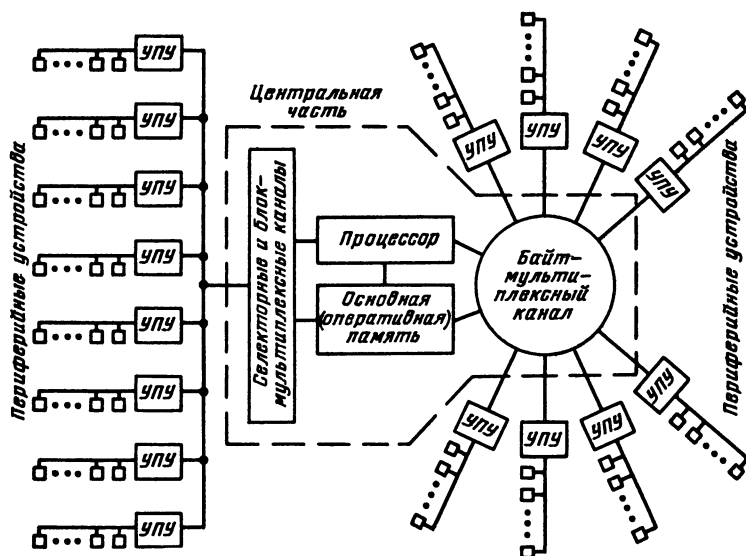


Рис. 1.4. Структура современной ЭВМ общего назначения (ЕС ЭВМ)

Современное представление о методах использования ЭВМ для научно-технических расчетов предполагает программирование на алгоритмических языках, ввод в машину текстов программ на этих языках и их преобразование, вывод результатов расчетов в виде оформленных таблиц с текстовыми надписями, хранение в машине больших программных массивов, в том числе разнообразных пакетов прикладных программ. В результате произошло стирание различий в требованиях к структуре ЭВМ и способам представления информации при решении указанных выше обобщенных типов задач — появились *ЭВМ общего назначения*, которые и в настоящее время выполняют большой объем вычислительных работ и машинной обработки информации в различных ВЦ и АСУ.

ЭВМ общего назначения имеет работающее с большой скоростью устройство обработки информации (процессор), память большой емкости, широкий набор периферийных устройств для ввода и вывода, хранения и отображения информации, гибкую систему команд и способ кодирования информации, учитывающие требования научно-технических расчетов и процессов обработки данных.

На рис. 1.4 представлена типичная структура ЭВМ общего назначения. Такую структуру, например, имеют машины ЕС ЭВМ.

Собственно обработка информации производится электронным процессором, содержащим арифметическо-логическое и управляющее устройства. В ЭВМ возникает проблема организации взаимодействия быстродействующего процессора с большим числом сравнительно медленно действующих периферийных устройств (ПУ).

Для эффективного использования технических средств необходима параллельная работа во времени процессора и периферийных устройств. Такой режим в машинах общего назначения организуется при помощи специализированных *процессоров ввода-вывода (каналов ввода-вывода)* информации. Периферийные устройства связываются с каналами через собственные блоки управления (УПУ) и систему сопряжения, называемую *интерфейсом ввода-вывода*.

Данные об основных характеристиках ЭВМ общего назначения приведены в § 1.7.

### **1.6.3. Малые ЭВМ**

Имеется большое число, условно говоря, «малых» применений вычислительных машин, таких, как автоматизация производственного контроля изделий, обработка данных при эксперимен-

тах, прием и обработка данных с линий связи, управление технологическими процессами, управление станками и разнообразными цифровыми терминалами (расчерчивающими устройствами и др.), малые расчетные инженерные задачи и т. д. Для этих применений ЭВМ общего назначения слишком велики и дороги.

Возникла необходимость в сравнительно небольших, простых, надежных и, главное, дешевых ЭВМ, в которых обеспечиваются простота программирования и наглядность системы программного обеспечения в отличие от сложности современных операционных систем ЭВМ общего назначения, а также сравнительно простая эксплуатационного обслуживания.

Развитие технологии интегральных микросхем позволило создать машины, удовлетворяющие указанным выше требованиям. Уменьшение объема аппаратуры и стоимости машин достигнуто, в первую очередь, за счет короткого машинного слова (16 разрядов вместо 32—64 в машинах общего назначения), уменьшения по сравнению с ЭВМ общего назначения количества типов обрабатываемых данных, ограниченного набора команд, сравнительно небольшой емкости оперативной памяти и небольшого набора периферийных устройств. Подобные машины за свои небольшие размеры получили название *малых* или *мини-ЭВМ*.

Для преодоления трудностей, возникающих при проектировании малых ЭВМ из-за короткого машинного слова, предложен ряд решений по представлению данных, адресации, составу и структуре команд, логической структуре процессора, организации обмена информацией между устройствами ЭВМ.

Типичная структура малой ЭВМ показана на рис. 1.5. Малые ЭВМ отличается наряду с другими особенностями более простая, чем у машин общего назначения, весьма гибкая структура, называемая *магистрально-модульной*, основу которой составляет общая магистраль (шина), к которой подсоединяются в нужных номенклатуре и количествах все устройства машины, выполняемые в виде конструктивно законченных модулей. Устройства машины обмениваются информацией через общую магистраль (шину).

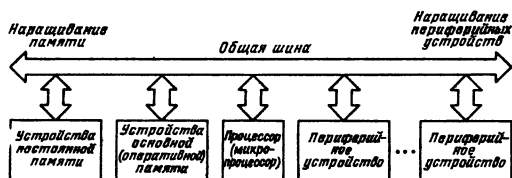


Рис. 1.5. Упрощенная структура малой и микроЭВМ



Такая структура оказывается эффективной, а система обмена данными через общую шину — достаточно динамичной лишь при сравнительно небольшом наборе периферийных устройств.

Универсальность применения при ограниченном наборе команд может быть обеспечена лишь при сравнительно высоком быстродействии машины — около 400—800 тыс. операций/с, что превышает скорость работы многих ЭВМ общего назначения. Высокое быстродействие должно позволить малым ЭВМ обслуживать технологические процессы в реальном масштабе времени, а также компенсировать замедление обработки данных, связанное с тем, что многие процедуры обработки при ограниченном объеме аппаратуры, скромном наборе команд и отсутствии специализации машины приходится реализовать не аппаратными средствами, а соответствующими подпрограммами (например, операции арифметики с плавающей точкой).

#### ***1.6.4. Микропроцессоры и микроЭВМ***

Положительный опыт разработки и применения малых ЭВМ оказал влияние на направление развития интегральной электроники. При переходе от схем с малой и средней степенями интеграции к интегральным микросхемам с большой и сверхбольшой степенями интеграции (БИС и СБИС) возникает проблема их применимости. Интегральную микросхему с большой степенью интеграции (БИС), содержащую тысячи логических элементов, не говоря о СБИС с ее десятками тысяч и более элементов, если это не схема памяти, трудно сделать пригодной для широкого круга потребителей. Первоначально считалось, что на основе автоматизированного проектирования будут выпускаться заказные БИС и СБИС, изготавливаемые по индивидуальным требованиям заказчиков. Однако в дальнейшем оказался возможным и другой путь — создание на одной или нескольких БИС или СБИС функционально законченного (8—16 разрядов и более) устройства обработки информации. Это устройство (микросхема или несколько образующих его микросхем) называют *микропроцессором*, так как оно по своим логическим функциям и структуре напоминает упрощенный вариант процессора обычных ЭВМ.

Микропроцессоры (МП) по быстродействию и возможностям системы команд приближаются к процессорам малых ЭВМ. Однако из-за ограниченного числа выводов корпуса МП (обычно 42) трудно реализовать интерфейс МП с внешним оборудованием с высокой пропускной способностью. В табл. 1.1 приведены характеристики некоторых микропроцессоров.

Устройство обработки данных, имеющее в своем составе один или несколько микропроцессоров, БИС постоянной и опера-

Т а б л и ц а 1.1. Характеристики микропроцессоров

Параметр	Значение параметров микропроцессора типа	
	К580	К1810
Число разрядов в слове	8	16
Скорость выполнения коротких операций (типа регистр—регистр), операций/с	500 тыс.	2 млн.
Скорость выполнения операций умножения и деления, операций/с	800 (по подпрограмме)	15 тыс.
Емкость адресуемой памяти, байт	До 64 К	До 1024 К
Число общих регистров	6	14
Число входов прерывания	1 (расширяется до 64)	1 (расширяется неограниченно)
Число адресуемых портов ввода-вывода	256	32 тыс.

тивной памяти, БИС управления вводом и выводом и др., называется *микроЭВМ*. МикроЭВМ оснащают необходимыми периферийными устройствами (см. рис. 1.5). Электронная аппаратура микроЭВМ содержит несколько десятков корпусов БИС и СИС, размещаемых на одной или нескольких съемных платах.

В микроЭВМ сочетаются высокая скорость выполнения операций в микропроцессоре, повышенная надежность, небольшая стоимость со сравнительно низкой пропускной способностью интерфейса, обусловленной ограничениями на число выводов корпусов БИС микропроцессора. Если по скорости выполнения операций микроЭВМ приближаются к современным малым ЭВМ, а по ряду эксплуатационных показателей (габаритные размеры, потребляемая мощность, надежность) они их превосходят, то из-за малой пропускной способности интерфейса и связанного с этим малого числа подключаемых ПУ применение микроЭВМ в настоящее время ограничивается системами с небольшим количеством источников и потребителей информации.

По этим же причинам затруднено использование микропроцессоров в качестве элементов при построении быстродействующих процессоров и каналов ввода-вывода ЭВМ общего назначения. Однако большие перспективы имеет применение микропроцессоров и микроЭВМ в периферийном оборудовании ЭВМ (устройствах управления дисками и лентами, дисплеях и других терминалах), в частности, для преобразования форматов данных, контроля, перекодирования, редактирования. При этом

расширяются логические функции ПУ, становится возможным создание так называемых *интеллектуальных терминалов*, выполняющих сложные процедуры предварительной обработки информации.

К микроЭВМ можно отнести такие широко распространенные вычислительные средства, как персональные компьютеры (см. § 1.8 и гл. 10).

Уникально малые, еще недавно показавшиеся бы фантастическими размеры микропроцессоров и микроЭВМ при достаточно больших вычислительных и логических возможностях, их дешевизна и высокая надежность представляют собой огромный качественный скачок в развитии вычислительной техники, расширивший практически беспредельно сферу использования цифровых вычислительных устройств и сделавший возможным создание на основе микропроцессоров, микроЭВМ, персональных компьютеров и микроконтроллеров, непосредственно встраиваемых в приборы, машины, оборудование и технологические процессы.

Микропроцессоры, микроЭВМ и персональные компьютеры открывают принципиально новые возможности для высокоэффективной автоматизации производственных процессов, научно-исследовательских и проектно-конструкторских работ, обработки информации при планировании и управлении производством на предприятиях во всех отраслях народного хозяйства. Поэтому с полным основанием создание и применение микропроцессоров, микроЭВМ и персональных компьютеров оцениваются как одно из важнейших направлений научно-технического прогресса.

### **1.6.5. МинисуперЭВМ и супермини-ЭВМ**

Выше отмечалось, что отсутствуют четкие границы между рассмотренными типами ЭВМ. В последнее время стали выделять два промежуточных типа ЭВМ: а) *минисуперЭВМ* и б) *супермини-ЭВМ (супермикроЭВМ)*.

МинисуперЭВМ — это упрощенные (в частности, за счет более короткого слова) многопроцессорные суперЭВМ, чаще всего со средствами векторной и конвейерной обработки (см. гл. 9), с высокой скоростью выполнения операций над числами с плавающей точкой.

К супермини-ЭВМ (супермикроЭВМ) относят высокопроизводительные (около 1 млн. операций/с и более) мини-ЭВМ, содержащие один процессор или небольшое число (четыре — восемь) слабо связанных процессоров, объединенных с общей памятью магистралью (общей шиной). Для супермини-ЭВМ характерно, что скорость выполнения его арифметических опера-

ций над числами с плавающей точкой существенно ниже скорости работы, определяемой по смеси команд, соответствующей информационно-логическим запросам.

### **1.6.6. Специализированные ЭВМ**

До сих пор речь шла об ЭВМ универсальных в том смысле, что в пределах своих вычислительных и логических возможностей они предназначены для использования практически в любых отраслях народного хозяйства.

Однако существуют такие области применения вычислительной техники, где специфические условия работы ЭВМ или специфические требования к ее характеристикам, особенности подлежащего реализации фиксированного набора алгоритмов делают невозможным или неэффективным применение универсальных серийно выпускаемых ЭВМ.

В подобных случаях может оказаться целесообразным создание специализированных ЭВМ и цифровых устройств, конструкция, архитектура и характеристики которых ориентированы на определенную, сравнительно узкую область применения. Примером являются бортовые ЭВМ систем управления аэрокосмическими объектами.

С появлением микропроцессоров облегчилось создание и расширились области эффективного применения специализированных цифровых устройств и машин.

## **1.7. Понятие о системах ЭВМ.**

### **Единая система ЭВМ общего назначения (ЕС ЭВМ) и система малых ЭВМ (СМ ЭВМ)**

Расширение сферы применения вычислительной техники и особенно ее использование в автоматизированных системах обработки информации в области планирования, экономики, учета привели к включению в состав машины большого комплекса разнообразных периферийных устройств для ввода информации, ее запоминания и хранения, регистрации и отображения. Конкретные применения предъявляют различные требования к составу периферийных устройств, объему оперативной и внешней памяти и т. п.

Это привело к тому, что при проектировании вычислительной техники концепцию вычислительной машины с фиксированным составом оборудования (где главное место занимало само устройство обработки информации) сменила концепция *агрегированной вычислительной машины (системы)* с переменным составом оборудования, который определяется выполняемыми

ею функциями. При таком подходе отдельные функциональные устройства выполняются в виде агрегатов (модулей), которые в нужных номенклатуре и количестве объединяются в ЭВМ. Одним из агрегатов такой системы оказывается устройство обработки информации — процессор.

Существенное место в агрегатированных машинах (системах) занимают унифицированные сопряжения (*интерфейсы*), обеспечивающие обмен информацией между агрегатами (модулями), входящими в состав ЭВМ, и допускающие подключение необходимого состава периферийных устройств.

Различные области применения ЭВМ предъявляют разнообразные требования и к самим устройствам обработки информации — к процессорам (в отношении длины машинного слова, скорости обработки информации и т. д.). Поэтому естественным развитием концепции агрегатированной вычислительной машины с переменным составом оборудования явилось создание *рядов или систем ЭВМ*, состоящих из информационно- и программно-совместимых машин, обладающих различными характеристиками.

*Информационная совместимость* ЭВМ предполагает единые способы кодирования информации и форматы данных или хотя бы по меньшей мере одинаковые или кратные длины машинных слов в различных моделях.

*Программная совместимость* означает, что программы, составленные для одной модели, могут выполняться на других моделях ряда. Практически это достигается единой для всех машин ряда системой команд. Единая система команд позволяет иметь общие для машин системы ЭВМ операционные системы.

Помимо программной совместимости машины, входящие в систему ЭВМ, должны обладать *аппаратурной совместимостью*, заключающейся в возможности присоединения к любой модели ЭВМ, точнее, к ее ядру, состоящему из процессора в оперативной памяти, любых ПУ, общих для всей системы ЭВМ (для ряда моделей).

Общие форматы данных и команд, единая система команд, общие операционные системы, общая номенклатура ПУ, общие методы технической и программистской эксплуатации и обслуживания (эксплуатационная совместимость), единая методика построения технической документации — вот что превращает совокупность отдельных моделей ЭВМ в систему.

Рассмотренные принципы построения систем ЭВМ полностью реализованы в Единой системе ЭВМ (ЕС ЭВМ) общего назначения и в значительной степени в Системе малых и микро-ЭВМ (СМ ЭВМ), созданных совместно странами, входящими в Совет Экономической Взаимопомощи. В нашей стране микро-

**Т а б л и ц а 1.2. Характеристики некоторых ЭВМ общего назначения Единой системы**

Модель ЕС ЭВМ	Производительность, тыс. операций/с	Максимальная емкость ОП, Кбайт	Байт-мультиплексные каналы		Блок-мультиплексные и селекторные каналы	
			Количество	Скорость передачи, Кбайт/с	Общее количество	Скорость передачи, Кбайт/с
ЕС-1035 (СССР, НРБ)	140	512	1	30	4	800
ЕС-1036 (СССР)	400	2048— 4096	1	40	4	1500
ЕС-1045 (СССР, ПНР)	800	4096	1—2	40	5	1300
ЕС-1046 (СССР)	1300	8192	2	50	4	1500, 3000
ЕС-1055 (ГДР)	600	2048	2	40	6	1500
ЕС-1065 (двух- процессорная) (СССР)	6000	16 384	2	200	6, 14	1500, 3000
ЕС-1066 (СССР)	5500	16 384	2	200	10	1500, 3000

Примечание. Производительность указана для набора операций, характерных для научно-технических расчетов.

ЭВМ создаются также помимо СМ ЭВМ еще и в рамках семейства «Электроника».

Типичная конфигурация модели ЕС ЭВМ показана на рис. 1.4, а общий вид ЭВМ ЕС-1045 дан на рис. 1.6.

Архитектура машин ЕС ЭВМ, первая очередь которых появилась в начале 70-х годов, в дальнейшем подвергалась существенной модернизации при разработке второй и третьей очередей ЕС ЭВМ, при этом осуществлялся переход к использованию более быстродействующих интегральных микросхем, в том числе схем средней и большей степеней интеграции (последние — главным образом для построения памяти), расширялись функциональные возможности машин (виртуальная память, система виртуальных машин, расширенная система команд, развитие систем автоматизированного контроля и диагностирования), повышалась пропускная способность каналов ввода-вывода, совершенствовались характеристики периферийных устройств, производительность процессоров старших моделей увеличилась до 5—6 млн. операций/с, возрастала емкость ОП, получили развитие системные средства для организации многомашинных и многопроцессорных систем.

Развитие ЕС ЭВМ осуществлялось с сохранением основных принципов архитектуры, чтобы имелась возможность использо-

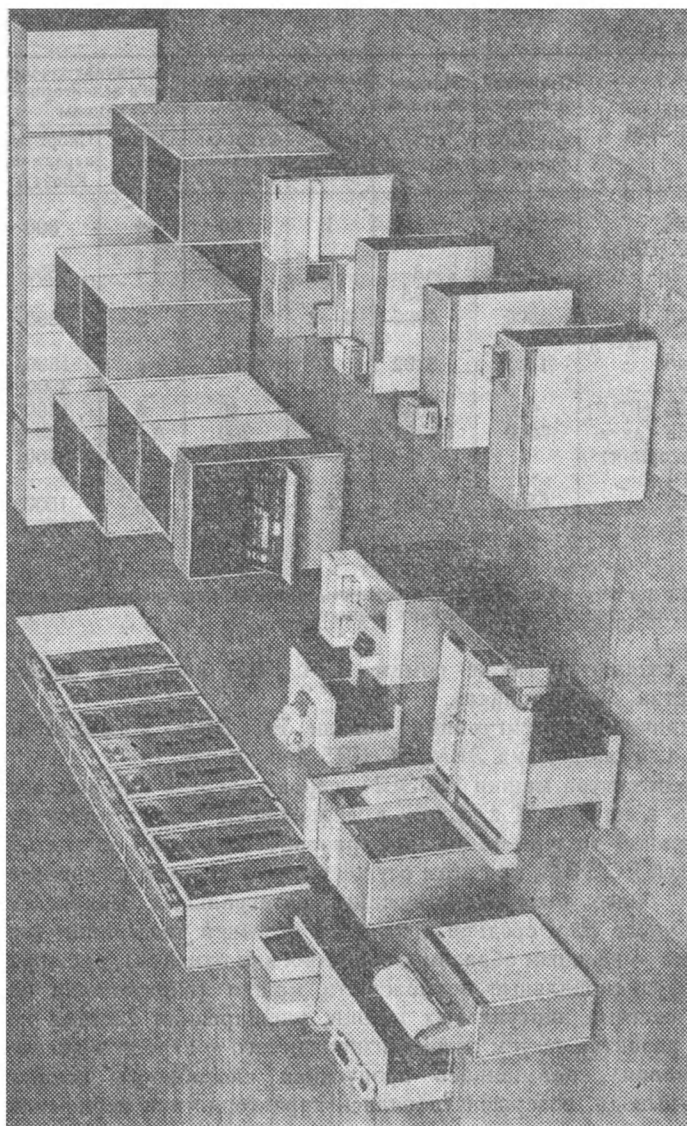


Рис. 1.6. Общий вид ЭВМ общего назначения

**Таблица 1.3. Характеристики малых и микроЭВМ СМ ЭВМ и семейства «Электроника»**

Модель	Длина слова, разрядов	Производительность, тыс. коротких операций/с	Максимальная емкость ОП, Кбайт	Используемые микропроцессорные средства
СМ-1420	16	1000	248	МП К1804
СМ-1300	16	150—500	56	МП К1802
СМ-1800	8	200	64	МП К580
СМ-1810	16	250	256	МП К1810
СМ-1700 (супермини-ЭВМ)	32	2800 (эквивалентных коротких операций) или 300 на программе «Вестоун»	1024—5120	МП К1804
«Электроника-60»	16	250	64	МП К581
Дисплейный вычислительный комплекс (ДВК) на базе микроЭВМ «Электроника НЦ80-01Д»	16	500	64	МП К1801

вания в новых моделях ранее созданного для машин ЕС программного обеспечения.

В табл. 1.2 приведены основные параметры некоторых машин общего назначения ЕС ЭВМ, выпускаемых промышленностью стран СЭВ [48, 49, 18, 45].

Характеристики некоторых моделей малых и микроЭВМ, входящих в состав СМ ЭВМ, а также в семейство «Электроника», приведены в табл. 1.3.

## 1.8. Персональные компьютеры

Успехи в развитии микропроцессоров и микроЭВМ ознаменовались созданием нового класса средств вычислительной техники — *персональных компьютеров* (ПК), предназначенных для индивидуального обслуживания пользователя в условиях непосредственного контакта с ним. Общий вид одной из моделей ПК представлен на рис. 1.7.

Персональные компьютеры обладают при небольших габаритных размерах и стоимости широкими вычислительными и логическими возможностями, удобными простыми в обращении



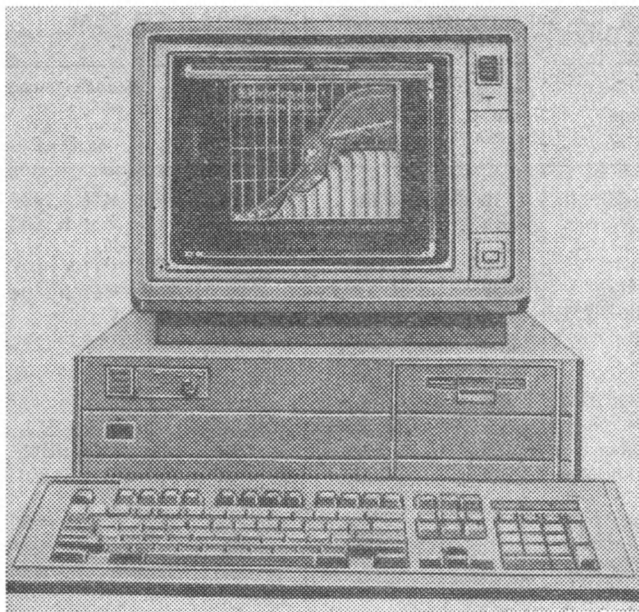


Рис. 1.7. Общий вид персонального компьютера

средствами для взаимодействия человека с компьютером, а, главное, снабжены ориентированным на пользователя, не имеющего специальной профессиональной подготовки, «дружественным» к нему обширным программным обеспечением, включающим в себя эффективные программные пакеты для выполняемых повседневно человеком процедур подготовки и редактирования текстов, вычислений, работы с базами данных и т. п., а также средства обучения работе с ПК, поддержки диалога пользователя с ПК, подсказки при ошибках в действиях пользователя. Общение пользователя с ПК становится более комфортным благодаря внесению программным обеспечением в этот процесс игровых компонентов. Сказанное объясняет, почему появление ПК ознаменовало революцию в технологии использования ЭВМ.

Вычислительные машины в форме ПК пришли на рабочие места ученых, конструкторов, технологов, руководителей разного ранга, сотрудников учреждений, журналистов и многих других работников, деятельность которых связана с получением, накоплением и обработкой информации.

Использование ПК во многих случаях позволяет повысить производительность и качество труда. Поэтому на основе ПК со-

здаются *автоматизированные рабочие места* (АРМ) работников разных профессий, в которых реализуются автоматизация проектирования, научных исследований, диспетчеризации производственных процессов и т. д.

Персональный компьютер пришел в наши квартиры как новая форма бытовой техники, используемая, в первую очередь, в учебных целях (компьютерные учебные курсы) и для досуга (компьютерные игры).

Появление ПК сделало актуальной и одновременно возможной компьютеризацию многих сторон нашей жизни. Для многих людей (в первую очередь, молодого поколения) стало необходимым овладение компьютерной грамотностью.

Существует и быстро пополняется множество различных моделей ПК. В новых моделях представляемые пользователю возможности быстро расширяются, в первую очередь, за счет увеличения производительности процессоров, емкостей основной и внешней памяти, повышения качества и гибкости электронной графики, качества печати и т. д.

Среди множества моделей ПК можно выделить следующие группы:

ПК для использования в быту (*бытовые ПК*) со сравнительно небольшими вычислительными ресурсами (производительность, емкость памяти), комплексируемые с обычными телевизорами;

*ПК для учебных классов* школ и других учебных заведений с умеренными вычислительными ресурсами, с собственными дисплейными средствами и средствами локальной сети (см. гл. 16) для объединения ПК учащихся и ПК педагога в учебный комплекс;

*профессиональные ПК* с обширными вычислительными ресурсами и эффективным набором периферийных средств (цветной графический дисплей, высококачественное печатающее устройство), предназначенные для поддержки профессиональной деятельности работников различных профессий, создания автоматизированных рабочих мест.

Характеристики некоторых моделей профессиональных персональных компьютеров приведены в гл. 10.

## 1.9. Классификация вычислительных систем

Стремление удовлетворить требования разнообразных областей и форм применения электронной вычислительной техники, повысить производительность и расширить логические возможности ЭВМ, повысить надежность их функционирования, облегчить контакты человека с ЭВМ при подготовке программ

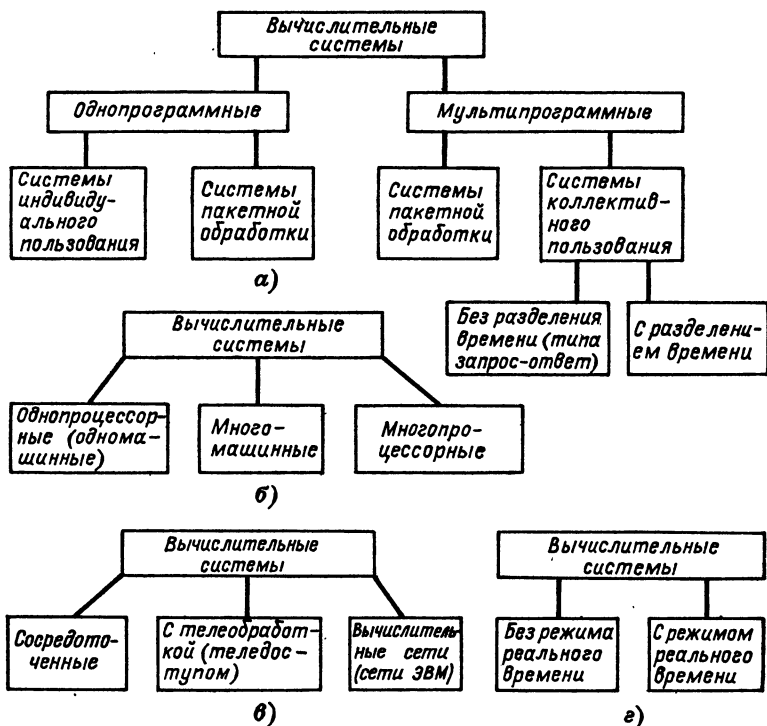


Рис. 1.8. Классификация вычислительных систем

и в процессе решения задач, повысить обслуживаемость машин привело в ряде случаев к созданию таких объектов вычислительной техники, которые из-за сложности входящего в них оборудования, тесной логической взаимосвязи аппаратурных и программных средств при реализации сложных процессов функционирования, множества возможных конфигураций, территориальной рассредоточенности оборудования не укладываются в наше представление о понятии машина. В таких случаях вместо термина *вычислительная машина* пользуются термином *вычислительная система*.

Одним из важнейших путей повышения производительности вычислительных машин и систем, их эффективности и надежности является использование различных форм параллелизма в функционировании вычислительного оборудования. Поэтому в основу классификации ВС следует положить в первую очередь реализуемую форму параллелизма.

По режиму работы ВС делятся на *однопрограммные* и *мультипрограммные* (рис. 1.8).

В однопрограммной системе в памяти машины присутствует только одна рабочая программа (или часть ее), которая, начав выполняться, завершается до конца. При этом, даже если допускается совмещение во времени операций ввода-вывода с обработкой данных, возможен простой оборудования (например, машина не может продолжать работу, пока не будут введены новые данные). Стремление повысить эффективность использования вычислительного оборудования привело к разработке мультипрограммных машин и систем, которые могут одновременно выполнять несколько программ или несколько частей одной и той же программы.

Когда говорится об одновременности выполнения программ, то подразумевается, что процессор после выполнения части одной программы может перейти к выполнению другой программы, не закончив ее, перейти к третьей и т. д., сохраняя возможность вернуться позднее к неоконченным программам и продолжить их выполнение. При этом моменты и очередность переключений программ должны быть выбраны так, чтобы повысить общую эффективность вычислительной системы, хотя время, в течение которого решается каждая отдельная задача, по сравнению со временем в однопрограммном режиме может даже увеличиться.

Мультипрограммный режим реализуется и по отношению к частям одной программы, что приводит к сокращению времени ее выполнения по сравнению со временем, когда режим мультипрограммирования не используется.

*Классификация систем по режиму обслуживания* (на рис. 1.8, а). Процесс решения задачи может рассматриваться как обслуживание пользователя ВС. Рассмотрим основные режимы обслуживания.

*Режим индивидуального пользования.* Машина предоставляется полностью в распоряжение пользователя, по крайней мере на время решения его задачи. Пользователь имеет непосредственный доступ к машине и может вводить информацию в оперативную память машины (или выводить из нее), используя устройства ввода-вывода.

Режим индивидуального пользования удобен пользователю, но в этом режиме плохо используется вычислительное оборудование из-за простоев, когда пользователь, получив некоторый промежуточный результат (например, при отладке программы), обдумывает, что он будет делать дальше. Этот режим применялся в ЭВМ первого поколения, а в настоящее время возродился как форма использования персональных компьютеров.

*Режим пакетной обработки.* В этом режиме пользователи не имеют непосредственного доступа к ВС. Подготовленные ими программы передаются персоналу, обслуживающему систему, и затем накапливаются во внешней памяти (на магнитных дисках, лентах и т. п.). Система последовательно либо по заранее составленному расписанию выполняет накопленный пакет программ.

Пакетная обработка может производиться в однопрограммном и мультипрограммном режимах.

Мультипрограммная пакетная обработка обеспечивает высокую степень загрузки вычислительного оборудования, но при этом из-за отсутствия непосредственной связи между системой и пользователем производительность и эффективность труда самих пользователей снижаются по сравнению с индивидуальным обслуживанием. Это противоречие преодолевается путем создания систем коллективного пользования, содержащих высокопроизводительные ЭВМ, или, наоборот, путем применения персональных ЭВМ умеренной производительности в режиме индивидуального пользования.

*Режим коллективного пользования или многопользовательский режим* — форма обслуживания, при которой возможен одновременный доступ нескольких независимых пользователей к вычислительным ресурсам мощной ВС. Каждому пользователю предоставляется терминал, с помощью которого он устанавливает связь с системой коллективного пользования (ВСКП).

В наиболее простых ВСКП обработка всех запросов занимает примерно одно и то же время (системы типа «запрос — ответ»).

Обеспечение более тесного взаимодействия пользователей с вычислительными средствами в системе коллективного пользования, в которой запросы сильно разнятся по времени их обработки, требует в первую очередь сокращения времени ожидания пользователем результата выполнения коротких программ (коротких запросов), для чего применяют различные методы квантования времени, уделяемого процессором для выполнения отдельных программ. Системы коллективного пользования с квантованным обслуживанием называются *системами с разделением времени*.

По количеству процессоров (машин) в ВС, определяющему возможность параллельной обработки программ, ВС делятся на *однопроцессорные (одномашинные)*, *многомашинные* и *многопроцессорные* (рис. 1.8, б). Многомашинные и многопроцессорные ВС создаются для повышения производительности и надежности вычислительных систем и комплексов.

По особенностям территориального размещения и организации взаимодействия частей системы различают следующие типы ВС (рис. 1.8, в).

*Сосредоточенные ВС.* В этих системах весь комплекс оборудования, включая терминалы пользователей, сосредоточен практически в одном месте и связь между отдельными машинами и устройствами комплекса обеспечивается, причем без существенных запаздываний, стандартными для системы внутренними интерфейсами (без использования передачи данных по каналам связи).

*Вычислительные системы с телеобработкой (или теледоступом).* В этих ВС отдельные источники и приемники информации, включая терминалы пользователей, расположены на таком значительном расстоянии от вычислительных средств, что связь их с центральными средствами ВС осуществляется путем передачи данных по каналам связи.

*Вычислительные сети (сети ЭВМ)* представляют собой территориально рассредоточенную многомашинную систему, состоящую из взаимодействующих ЭВМ, связанных между собой каналами передачи данных.

По особенностям функционирования ВС во времени различают ВС, работающие не в масштабе реального времени, и *ВС реального времени* (рис. 1.8, г). Последние должны работать в темпе с процессом, информация о котором автоматически поступает в ВС и обрабатывается.

Результаты обработки информации должны получаться столь быстро, чтобы можно было ими воспользоваться для воздействия на сам процесс.

Вычислительные системы различаются по назначению следующим образом:

- информационно-справочные;

- сбора и обработки данных в автоматизированных системах, в частности информационно-планирующие;

- управления технологическими процессами в реальном времени;

- автоматизации обработки данных при сложных экспериментах;

- автоматизации проектирования и др.

Многие применения ВС, например автоматизация проектирования, требуют, чтобы поиск решений задач осуществлялся в режиме взаимодействия человека с машиной.

Проблемная ориентация ВС достигается соответствующей конфигурацией аппаратурных средств и укомплектованием их соответствующим проблемно-ориентированным программным обеспечением.

## Контрольные вопросы

1. Каковы основные принципы построения ЭВМ?  
В чем различие фоннеймановской и гарвардской моделей ЭВМ?
2. Что такое архитектура ЭВМ?
3. Чем определяется поколение ЭВМ? Какие существовали поколения ЭВМ в процессе развития этих машин? К какому поколению относится ЭВМ, на которой Вы работаете?
4. Назовите основные типы ЭВМ (вычислительных средств) и их области применения. В чем различие ЭВМ общего назначения и малых ЭВМ?
5. Назовите основные характеристики ЭВМ. Что такое «смесь команд» и как она используется при определении производительности ЭВМ?
6. Почему развитие электронной вычислительной техники, в том числе микропроцессорных средств, персональных компьютеров, супер-ЭВМ, является приоритетным направлением научно-технического прогресса?
7. С какой целью создаются многомашинные и многопроцессорные вычислительные системы и комплексы?
8. В чем состоит особенность работы ЭВМ в масштабе реального времени?

## Глава 2

# ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ЭВМ

## 2.1. Позиционные системы счисления

Под *системой счисления* понимается способ представления любого числа с помощью некоторого алфавита символов, называемых цифрами. Существуют различные системы счисления. От их особенностей зависят наглядность представления числа при помощи цифр и сложность выполнения арифметических операций.

Римская система счисления является примером системы с очень сложным способом записи чисел и громоздкими правилами выполнения арифметических операций.

Наглядность представления чисел и сравнительная простота выполнения арифметических операций характерны для *позиционных систем счисления*.

Система счисления называется позиционной, если одна и та же цифра имеет различное значение, определяющееся позицией цифры в последовательности цифр, изображающей число. Это

значение меняется в однозначной зависимости от позиции, занимаемой цифрой, по некоторому закону. Позиционной является десятичная система, используемая в повседневной жизни. Помимо десятичной существуют другие позиционные системы. Некоторые из них нашли применение в вычислительной технике.

Количество  $s$  различных цифр, употребляемых в позиционной системе, называется ее основанием. Эти цифры обозначают  $s$  целых чисел, обычно  $0, 1, \dots, (s-1)$ . В десятичной системе используются десять цифр:  $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$ ; эта система имеет основанием число десять.

В общем случае в позиционной системе с основанием  $s$  любое число  $x$  может быть представлено в виде полинома от основания  $s$ :

$$x = e_r s^r + e_{r-1} s^{r-1} + \dots + e_1 s^1 + e_0 s^0 + e_{-1} s^{-1} + e_{-2} s^{-2} + \dots, \quad (2.1)$$

где в качестве коэффициентов  $e_i$  могут стоять любые из  $s$  цифр, используемых в системе счисления.

Принято представлять числа в виде соответствующей (2.1) последовательности цифр:

$$x = e_r e_{r-1} \dots e_1 e_0 \cdot e_{-1} e_{-2} \dots$$

В этой последовательности точка (запятая) <sup>1</sup> отделяет целую часть числа от дробной (коэффициенты при положительных степенях, включая нуль, от коэффициентов при отрицательных степенях). Точка опускается, если нет отрицательных степеней. Позиции цифр, отсчитываемые от точки, называют разрядами. В позиционной системе счисления значение каждого разряда больше значения соседнего справа разряда в число раз, равное основанию  $s$  системы.

В ЭВМ применяют позиционные системы счисления с недесятичным основанием: двоичную, шестнадцатеричную, восьмеричную и др. В дальнейшем для обозначения используемой системы счисления будем заключать число в скобки и в индексе указывать основание системы счисления.

Наибольшее распространение в ЭВМ имеет *двоичная система* счисления. В этой системе используются только две («двоичные») цифры:  $0$  и  $1$ .

В двоичной системе любое число может быть представлено

---

<sup>1</sup> В этой книге целую часть числа от дробной будем отделять точкой, как это широко принято в литературе по вычислительной технике и программированию.



последовательностью двоичных цифр

$$x = \alpha_m \alpha_{m-1} \dots \alpha_1 \alpha_0 . \alpha_{-1} \alpha_{-2} \dots,$$

где  $\alpha_i$  либо 0, либо 1.

Эта запись соответствует сумме степеней числа 2, взятых с указанными в ней коэффициентами:

$$x = \alpha_m \cdot 2^m + \alpha_{m-1} \cdot 2^{m-1} + \dots \\ \dots + \alpha_1 \cdot 2^1 + \alpha_0 \cdot 2^0 + \alpha_{-1} \cdot 2^{-1} + \alpha_{-2} \cdot 2^{-2} + \dots \quad (2.2)$$

Например, двоичное число

$$(10101101.101)_2 = 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + \\ + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3},$$

как следует из приведенного разложения его по степеням числа 2, соответствует десятичному числу

$$(173.625)_{10}.$$

Двоичное изображение числа требует большего (для много-разрядного числа примерно в 3,3 раза) числа разрядов, чем его десятичное представление. Тем не менее применение двоичной системы создает большие удобства для проектирования ЭВМ, так как для представления в машине разряда двоичного числа может быть использован любой простой элемент, имеющий всего два устойчивых состояния. Такими элементами, например, являются триггерные схемы и т. п. Другим важным достоинством двоичной системы является простота двоичной арифметики.

В *восьмеричной системе* употребляются восемь цифр: 0, 1, 2, 3, 4, 5, 6, 7. Любое число в восьмеричной системе представляется последовательностью цифр

$$x = \beta_q \beta_{q-1} \dots \beta_1 \beta_0 . \beta_{-1} \beta_{-2} \dots,$$

в которой  $\beta_i$  могут принимать значения от 0 до 7.

Например, восьмеричное число

$$(703.04)_8 = 7 \cdot 8^2 + 0 \cdot 8^2 + 3 \cdot 8^0 + 0 \cdot 8^{-1} + 4 \cdot 8^{-2} = (451.0625)_{10}.$$

В *шестнадцатеричной системе* для изображения чисел употребляются 16 цифр: от 0 до 15, при этом, чтобы одну цифру не изображать двумя знаками, приходится вводить специальные обозначения для цифр, больших девяти. Обозначим первые десять цифр этой системы цифрами от 0 до 9, а старшие пять цифр — латинскими буквами: 10 — A, 11 — B, 12 — C, 13 — D, 14 — E, 15 — F.

Например, шестнадцатеричное число

$$(B2E.4)_{16} = 11 \cdot 16^2 + 2 \cdot 16^1 + 14 \cdot 16^0 + 4 \cdot 16^{-1} = (2862.25)_{10}.$$

Общие методы перевода чисел из одной системы счисления в другую изложены в [53]. Здесь ограничимся только рассмотрением правил преобразования восьмеричных и шестнадцатеричных чисел в двоичные и наоборот. Эти правила исключительно просты, поскольку основания восьмеричной и шестнадцатеричной систем есть целые степени числа два:  $8=2^3$ ,  $16=2^4$ .

Для перевода восьмеричного (шестнадцатеричного) числа в двоичную форму достаточно заменить каждую цифру этого числа соответствующим трехразрядным (четырёхразрядным) двоичным числом, при этом отбрасывают ненужные нули в старших разрядах, например

$$\left( \underbrace{3}_{011} \quad \underbrace{0}_{000} \quad \underbrace{5}_{101} \cdot \underbrace{4}_{100} \right)_8 = (11000101.100)_2;$$

$$\left( \underbrace{7}_{0111} \quad \underbrace{B}_{1011} \quad \underbrace{2}_{0010} \cdot \underbrace{E}_{1110} \right)_{16} = (11110110010.1110)_2.$$

Для перехода от двоичной к восьмеричной (или шестнадцатеричной) системе поступают следующим образом: двигаясь от точки влево и вправо, разбивают двоичное число на группы по три (четыре) разряда, дополняя при необходимости нулями крайние левую и правую группы. Затем группу из трех (четырёх) разрядов заменяют соответствующей восьмеричной (шестнадцатеричной) цифрой.

Приведем примеры:

а) перевод двоичного числа 1101111001.1101 в восьмеричное:

$$\underbrace{001}_1 \quad \underbrace{101}_5 \quad \underbrace{111}_7 \quad \underbrace{001}_1 \cdot \underbrace{110}_6 \quad \underbrace{100}_4 = (1571.64)_8;$$

б) перевод двоичного числа 11111111011, 100111 в шестнадцатеричное:

$$\underbrace{0111}_7 \quad \underbrace{1111}_F \quad \underbrace{1011}_B \cdot \underbrace{1001}_9 \quad \underbrace{1100}_C = (7FB.9C)_{16}.$$

В настоящее время в большинстве ЭВМ используются двоичная система и двоичный алфавит для представления и хранения чисел, команд и другой информации, а также при выполнении арифметических и логических операций.

Шестнадцатеричная и восьмеричная системы применяются в текстах программ для более короткой и удобной записи двоичных кодов команд, адресов и операндов. Кроме того, эти системы применяются в ЭВМ при некоторых формах представления чисел (см. § 2.3).

## 2.2. Двоичная арифметика

Правила выполнения арифметических действий над двоичными числами задаются таблицами двоичного сложения, вычитания и умножения.

Таблица двоичного сложения	Таблица двоичного вычитания	Таблица двоичного умножения
$0+0=0$	$0-0=0$	$0\times 0=0$
$0+1=1$	$1-0=1$	$0\times 1=0$
$1+0=1$	$1-1=0$	$1\times 0=0$
$1+1=0 + \text{единица переноса в старший разряд}$	$10-1=1$	$1\times 1=1$

При сложении двоичных чисел в каждом разряде в соответствии с таблицей двоичного сложения производится сложение двух цифр слагаемых или двух этих цифр и 1, если имеется перенос из соседнего младшего разряда. В результате получается цифра соответствующего разряда суммы и, возможно, также 1 переноса в старший разряд. Приведем пример сложения двух двоичных чисел:

Переносы

$$\begin{array}{r}
 \phantom{1} 111 \\
 + \phantom{1} 110111.01 \\
 \phantom{1} 10011.10 \\
 \hline
 1001010.11
 \end{array}
 \qquad
 \begin{array}{r}
 \phantom{1} 1 \\
 + \phantom{1} 55.25 \\
 \phantom{1} 19.5 \\
 \hline
 74.75
 \end{array}$$

Справа показано сложение тех же чисел, представленных в десятичной системе.

При вычитании двоичных чисел в данном разряде при необходимости занимается 1 из следующего, старшего разряда. Эта занимаемая 1 равна двум 1 данного разряда. Заем производится каждый раз, когда цифра в разряде вычитаемого больше цифры в том же разряде уменьшаемого. Поясним сказанное примером:

$$\begin{array}{r}
 \phantom{1} 11011.10 \\
 - \phantom{1} 1101.01 \\
 \hline
 1110.01
 \end{array}$$

Умножение двоичных многоразрядных чисел производится путем образования частичных произведений и последующего их суммирования. В соответствии с таблицей двоичного умножения

каждое частичное произведение равно 0, если в соответствующем разряде множителя стоит 0, или равно множимому, сдвинутому на соответствующее число разрядов влево, если в разряде множителя стоит 1. Таким образом, операция умножения много-разрядных двоичных чисел сводится к операциям сдвига и сложения. Положение запятой определяется так же, как и при умножении десятичных чисел. Сказанное поясняется примером:

$$\begin{array}{r}
 1011.1 \times 101.01 = 111100.011 \\
 \times \begin{array}{r} 10111 \\ 10101 \end{array} \\
 \hline
 \begin{array}{r} 10111 \\ 00000 \\ + 10111 \\ 00000 \\ \hline 10111 \\ 111100011 \end{array}
 \end{array}$$

Особенности выполнения деления двоичных чисел поясняются следующим примером:

$$\begin{array}{r}
 1100.011 : 101.01 = ? \\
 \begin{array}{r} 1100011 \overline{)10010} \\ - 10010 \\ \hline 11011 \\ - 10010 \\ \hline 10010 \\ - 10010 \\ \hline 00000 \end{array}
 \end{array}$$

Благодаря простоте правил двоичного сложения, вычитания и умножения применение в ЭВМ двоичной системы счисления позволяет упростить схемы устройств, выполняющих арифметические операции.

### 2.3. Прямой, обратный и дополнительный коды

В ЭВМ в целях упрощения выполнения арифметических операций применяют специальные коды для представления чисел. При помощи этих кодов упрощается определение знака результата операции. Операция вычитания (или алгебраического сложения) чисел сводится к арифметическому сложению кодов, облегчается выработка признаков переполнения разряд-

ной сетки. В результате упрощаются устройства ЭВМ, выполняющие арифметические операции.

Для представления чисел со знаком в ЭВМ применяют прямой, обратный и дополнительный коды.

Общая идея построения кодов такова. Код трактуется как число без знака, а диапазон представляемых кодами чисел без знака разбивается на два поддиапазона. Один из них представляет положительные числа, а другой — отрицательные. Разбиение выполняется таким образом, чтобы принадлежность к поддиапазону определялась максимально просто. Очень удобно формировать коды так, чтобы значение старшего разряда указывало на знак представляемых чисел. Использование такого кодирования позволяет говорить о старшем разряде как о знаковым и об остальных как о цифровых разрядах кода, хотя в общем код трактуется как число без знака.

*Прямой код* двоичного числа  $G$ , представляемого в  $n$ -разрядной сетке, определяется как

$$G_{\text{пр}} = \begin{cases} G & \text{при } G \geq 0; \\ A + |G| & \text{при } G \leq 0, \end{cases} \quad (2.3)$$

где  $A$  — величина, равная весу старшего разряда сетки (для дробей  $A = 1$ , а для целых  $A = 2^{n-1}$ ). Диапазон представляемых прямым кодом чисел  $0 \leq G < A$ .

Из определения следует, что положительные числа представляются кодами (числами без знака)  $0 \leq G_{\text{пр}} < A$ , а отрицательные  $A \leq G_{\text{пр}} < 2A$ . Признаком представления положительных (отрицательных) чисел является наличие 0 (1) в старшем разряде, называемом знаковым. Цифровые разряды прямого кода содержат модуль представляемого числа, что обеспечивает наглядность представления чисел в прямом коде.

Сложение в прямом коде чисел, имеющих одинаковые знаки, выполняется достаточно просто. Числа складываются, и сумме присваивается код знака слагаемых. Значительно более сложной является операция алгебраического сложения в прямом коде чисел с различными знаками. В этом случае приходится определять большее по модулю число, производить вычитание чисел и присваивать разности знак большего по модулю числа.

Операция вычитания (алгебраического сложения) сводится к операции простого арифметического сложения при помощи обратного и дополнительного кодов, используемых для представления в машине чисел со знаком.

*Обратный код* двоичного числа  $G$ , представляемого в  $n$ -разрядной сетке, определяется как

$$G_{\text{обр}} = \begin{cases} G & \text{при } G \geq 0; \\ B - |G| & \text{при } G \leq 0, \end{cases} \quad (2.4)$$

где  $B$  — величина наибольшего числа без знака, размещающегося в  $n$ -разрядной сетке (для дробей  $B=2-2^{-(n-1)}$ , а для целых  $B=2^n-1$ ). Диапазон представляемых обратным кодом чисел такой же, как и у прямых кодов:  $0 \leq |G| < A$ . По определению обратный код отрицательного числа представляет собой дополнение модуля исходного числа до наибольшего числа без знака, помещающегося в разрядную сетку. В связи с этим получение обратного кода двоичного отрицательного числа сводится к получению инверсии  $n$ -разрядного кода модуля этого числа. Так как модуль чисел, представимых в  $n$ -разрядной сетке,  $|G| < A$ , в старшем (знаковом) разряде обратного кода у положительных чисел будет 0, а у отрицательных 1. В цифровых разрядах обратного двоичного кода представляется либо модуль числа (для положительных чисел), либо его инверсия (для отрицательных чисел).

Дополнительный код двоичного числа  $G$ , представляемого в  $n$ -разрядной сетке, определяется как

$$G_{\text{доп}} = \begin{cases} G & \text{при } G \geq 0; \\ C - |G| & \text{при } G < 0, \end{cases} \quad (2.5)$$

где  $C$  — величина, равная весу разряда, следующего за старшим разрядом используемой разрядной сетки (для дробей  $C=2$ , а для целых чисел  $C=2^n$ ). Диапазон представляемых дополнительным кодом чисел отличается от диапазона прямого или обратного кода. Для положительных и отрицательных чисел поддиапазоны различны. Для положительных чисел (как и у прямого кода)  $0 \leq G < A$ , а для отрицательных  $0 < |G| \leq A$ . Из определения дополнительного кода следует, что старший (знаковый) разряд у кода положительного числа равен 0, а у кода отрицательного числа 1. В цифровых разрядах дополнительного кода положительного числа представляется модуль этого числа. Дополнительный код отрицательного числа удобно получать через обратный код.

Если рассматривать обратный и дополнительный коды числа как двоичные числа без знаков, то для отрицательных двоичных дробей  $G_{\text{доп}} = G_{\text{обр}} + 2^{-(n-1)}$ , а для отрицательных двоичных целых чисел  $G_{\text{доп}} = G_{\text{обр}} + 1$ .

Таким образом, дополнительный код отрицательного числа может быть получен из обратного путем прибавления 1 к младшему разряду обратного кода.

При выполнении расчетов на машине может возникнуть как «положительный», так и «отрицательный» 0. Положительный 0 в прямом, дополнительном и обратном кодах имеет вид

$$(+0)_{\text{пр}} = 000 \dots 0;$$

$$(+0)_{\text{доп}} = 000 \dots 0;$$

$$(+0)_{\text{обр}} = 000 \dots 0.$$

Отрицательный 0 изображается:  
в прямом коде

$$(-0)_{\text{пр}} = 100 \dots 0,$$

в обратном

$$(-0)_{\text{обр}} = 111 \dots 1;$$

в дополнительном коде отрицательный 0 отсутствует.

При представлении чисел дополнительным кодом 0 имеет единственное изображение. При применении обратного кода «положительный» и «отрицательный» 0 имеют разные изображения. Изменение знака числа, представленного в прямом коде, выполняется инвертированием его знакового разряда.

Изменению знака числа соответствует инвертирование его кода, если число представлено в обратном коде, и инвертирование и добавление 1 младшего разряда, если число представлено в дополнительном коде. В результате получается код соответствующего положительного числа.

Сказанное следует из (2.3) — (2.5).

Рассмотрим применение обратного и дополнительного кодов при алгебраическом сложении  $n$ -разрядных двоичных чисел  $G$  и  $Q$ . Могут быть сформулированы следующие правила (предполагаем, что модуль алгебраической суммы меньше 1 для дробей и меньше  $2^{n-1}$  для целых, и, следовательно, код суммы представим в  $n$ -разрядной сетке).

При алгебраическом сложении двух двоичных чисел, представленных обратным (или дополнительным) кодом, производится арифметическое суммирование этих кодов, включая разряды знаков. При возникновении переноса из разряда знака единица переноса прибавляется к младшему разряду суммы кодов<sup>1</sup> при использовании обратного кода и отбрасывается при использовании дополнительного кода. В результате получается алгебраическая сумма в обратном (дополнительном) коде.

*Признак переполнения разрядной сетки.* При алгебраическом сложении двух чисел, помещающихся в разрядную сетку, может возникнуть переполнение, т. е. может образоваться сумма, требующая для своего представления на один цифровой разряд больше по сравнению с разрядной сеткой слагаемых.

<sup>1</sup> Такой перенос называется круговым или циклическим.

Можно сформулировать следующее правило (признак) для обнаружения переполнения разрядной сетки.

При алгебраическом сложении двух двоичных чисел с использованием дополнительного (обратного) кода для их представления признаком переполнения является наличие переноса в знаковый разряд суммы при отсутствии переноса из ее знакового разряда (положительное переполнение) или наличие переноса из знакового разряда суммы при отсутствии переноса в ее знаковый разряд (отрицательное переполнение). Если и в знаковый, и из знакового разряда суммы есть переносы или нет этих переносов, то переполнение отсутствует. При положительном переполнении результат операции положительный, а при отрицательном — отрицательный.

Справедливость этого утверждения доказывается, например, в [22а].

Помимо рассмотренных кодов для представления чисел со знаком применяется еще *смещенный код*. Этот код обычно используется для представления целых чисел, задающих порядки чисел с плавающей точкой (см. § 2.4). Определяется смещенный код двоичного числа  $G$ , представленного в  $n$ -разрядной сетке, как

$$G_{\text{см}} = A + G, \quad (2.6)$$

либо, что то же самое,

$$G_{\text{см}} = \begin{cases} A + |G| & \text{при } G \geq 0; \\ A - |G| & \text{при } G < 0, \end{cases} \quad (2.6a)$$

где  $A$  (смещение) — величина, равная весу старшего разряда сетки (для целых  $A = 2^{n-1}$ ). Диапазон представляемых чисел такой же, как и у дополнительного кода. Единица в знаковом разряде смещенного кода указывает на представление положительного числа, нуль — отрицательного.

Цифровые разряды смещенного кода для положительного числа представляют модуль этого числа, для отрицательного — инверсию модуля, к которой подсуммирована 1 к младшему разряду.

Существует простое правило перехода к смещенному коду от дополнительного (и наоборот): для перехода необходимо инвертировать знаковый разряд кода.

Важной особенностью смещенного кода является то, что из  $G_{1\text{см}} > G_{2\text{см}}$  следует  $G_1 > G_2$ . Использование смещенных кодов упрощает сравнение чисел со знаком, сводя его к сравнению представляющих их чисел без знака (смещенных кодов).

Сложение смещенных кодов может выполняться в соответствии со следующим утверждением.



Сумма смещенных кодов с инвертированным старшим (знаковым) разрядом дает смещенный код суммы, при этом признаком получения положительного (отрицательного) переполнения является наличие (отсутствие) переноса из знакового разряда суммы, имевшего до инверсии значение 1 (0).

Вычитание смещенных кодов сводится к сложению уменьшаемого и вычитаемого, представляемого с противоположным знаком. Изменение знака числа в смещенном коде выполняется так же, как и в дополнительном.

В последнее время для представления порядков чисел с плавающей точкой стал применяться смещенный код с отрицательным нулем [35]. Это величина на единицу меньше, чем  $G_{\text{см}}$ :

$$G'_{\text{см}} = A - 1 + |G|,$$

или

$$G'_{\text{см}} = \begin{cases} A - 1 + |G| & \text{при } G \geq 0; \\ A - 1 - |G| & \text{при } G \leq 0. \end{cases} \quad (2.7a)$$

Диапазон представляемых чисел здесь иной: для положительных чисел  $1 \leq G \leq A$ , а для отрицательных  $0 \leq |G| \leq A - 1$ .

Сложение смещенных кодов с отрицательным нулем аналогично сложению смещенных кодов с положительным нулем, но требует дополнительного подсуммирования единицы:  $(G + Q)'_{\text{см}} = (G)'_{\text{см}} + (Q)'_{\text{см}} + 1$ . Вычитание выполняется как  $(G - Q)'_{\text{см}} = (G)_{\text{см}} + \overline{(Q)'_{\text{см}}}$ .

Признаки переполнения разрядной сетки те же, что были зафиксированы для смещенных кодов с положительным нулем.

## 2.4. Формы представления чисел в ЭВМ

Любая информация (числа, команды, алфавитно-цифровые записи и т. п.) представляется в ЭВМ в виде двоичных кодов (двоичных слов) фиксированной или переменной длины. Отдельные элементы двоичного кода, имеющие значение 0 или 1, называют разрядами или битами. В ЭВМ слова часто разбивают на части, называемые *слогами* или *байтами*. В современных ЭВМ широко используется байт, содержащий 8 бит (разрядов).

Двоичный разряд представляется в ЭВМ некоторым техническим устройством, например триггером, двум различным состояниям которого приписывают значения 0 и 1. Набор соответствующего количества таких устройств служит для представления многоразрядного двоичного числа (слова).

В ЭВМ применяют две формы представления чисел: с *фиксированной точкой* и с *плавающей точкой*. Эти формы называют также соответственно естественной и полулогарифмической.

При представлении чисел с фиксированной точкой положение точки фиксируется в определенном месте относительно разрядов числа. Обычно подразумевается, что точка находится или перед старшим цифровым разрядом, или после младшего. В первом случае могут быть представлены только числа, которые по модулю меньше 1, во втором — только целые числа.

На рис. 2.1 показаны примеры форматов данных для представления двоичных чисел с фиксированной точкой и соответствующие разрядные сетки. По сложившейся в вычислительной технике традиции нумерации разрядов (бит) в разрядной сетке в машинах общего назначения (ЕС ЭВМ) ведется слева направо, а в малых ЭВМ, микроЭВМ и микропроцессорах — справа налево. На разрядной сетке указаны веса разрядов.

При представлении числа со знаком для кода знака выделяется «знаковый» разряд (обычно крайний слева). В этом разряде 0 соответствует плюсу, 1 — минусу.

На рис. 2.1, а показан формат для чисел с точкой, фиксированной перед старшим цифровым разрядом. В этом формате могут быть с точностью до  $2^{-(n-1)}$  представлены числа (правильные дроби). Если для представления чисел используется прямой код, то они могут принимать значения в диапазоне

$$2^{-(n-1)} \leq |x| \leq 1 - 2^{-(n-1)}.$$

Первые ЭВМ были машинами с фиксированной точкой, причем точка фиксировалась перед старшим цифровым разрядом числа. В настоящее время, как правило, форму с фиксированной точкой применяют для представления целых чисел (точка фиксирована после младшего разряда).

Используют два варианта представления целых чисел: со знаком и без знака. В последнем случае все разряды разрядной сетки служат для представления модуля числа. В ЕС ЭВМ применяются оба указанных варианта представления целых чисел, причем каждый из вариантов реализуется как в формате 32-разрядного машинного слова этих машин, так и в формате 16-разрядного полуслова (рис. 2.1, б — д). Кроме того, используется формат 64-разрядного целого числа со знаком для представления делимого и произведения (на рис. 2.1 не показан).

Если точка фиксирована справа от младшего разряда, то в  $n$ -разрядной сетке целых чисел со знаком можно представлять нуль и положительные и отрицательные целые двоичные числа.

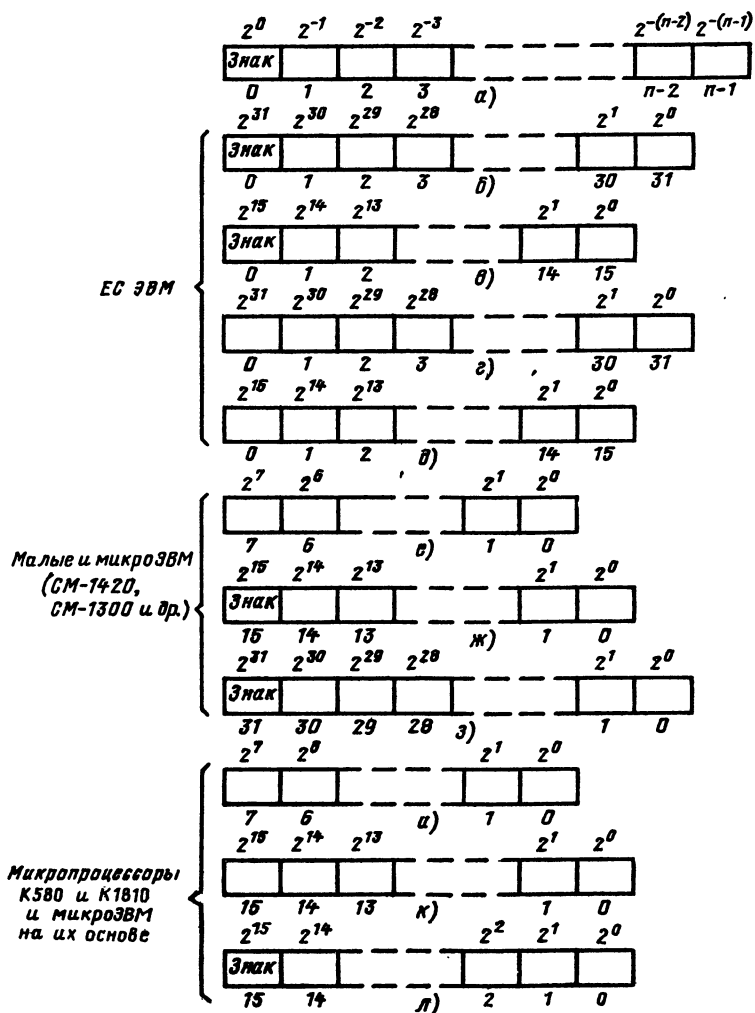


Рис. 2.1. Форматы данных для представления двоичных чисел с фиксированной точкой:

*а* — точка фиксирована перед старшим разрядом; *б, в* — целые числа со знаком в формате слова и полуслова (ЕС ЭВМ); *г, д* — целые числа без знака в формате слова и полуслова (ЕС ЭВМ); *е—з* — целые числа без знака в формате байта и со знаком в формате 16-разрядного слова (малые и микроЭВМ СМ-1420, СМ-1300 и др.); *и—л* — целые числа без знака и со знаком (микропроцессоры К580 и К1810 и микроЭВМ на их основе)

Дополнительный код позволяет использовать числа в диапазоне  $-2^{n-1} \leq x \leq 2^{n-1} - 1$ ,

что при  $n=32$  соответствует диапазону абсолютных десятичных целых чисел примерно от 1 до  $10^9$ .

В 16-разрядных малых и микроЭВМ (СМ-1420, СМ-1300 и др.) целые числа без знака могут быть представлены в формате 8-разрядного слова, числа со знаком — в формате 16-разрядного слова, а при некоторых операциях — в виде 32-разрядного слова (рис. 2.1,  $e-z$ ). Микропроцессоры 8-разрядные К580, 16-разрядные К1810 и микроЭВМ на их основе оперируют с целыми числами без знака в формате 8- и 16-разрядного слова, а К1810 кроме того и числами со знаком в формате 16-разрядного слова.

Представление чисел с фиксированной точкой используется как основное и единственное лишь в сравнительно небольших по своим вычислительным возможностям машинах, применяемых в системах передачи данных, для управления технологическими процессами и обработки измерительной информации в реальном масштабе времени.

В машинах, предназначенных для решения широкого круга вычислительных задач, основным является представление чисел с плавающей точкой, не требующее масштабирования данных. Однако в таких машинах часто наряду с этой формой представления чисел используется также и представление с фиксированной точкой для целых чисел, поскольку операции с такими числами выполняются за меньшее время. В частности, к операциям с целыми числами сводятся операции над кодами адресов (операции индексной арифметики).

Представление числа с плавающей точкой в общем случае имеет вид <sup>1</sup>

$$x = s^p q,$$

где  $q$  — мантисса числа  $x$ ;  $s^p$  — характеристика числа  $x$ ;  $p$  — порядок;  $s$  — основание характеристики (обычно целая степень числа 2).

Мантисса (дробь со знаком) и порядок (целое число со знаком) представляются в системе счисления с основанием, равным  $s$  (в соответствующей двоично-кодированной форме). Знак числа совпадает со знаком мантиссы.

Порядок  $p$ , который может быть положительным или отрицательным целым числом, определяет положение точки в числе  $x$ .

---

<sup>1</sup> Это представление числа называют также полулогарифмическим, так как часть числа — характеристика — выражена в логарифмической форме.

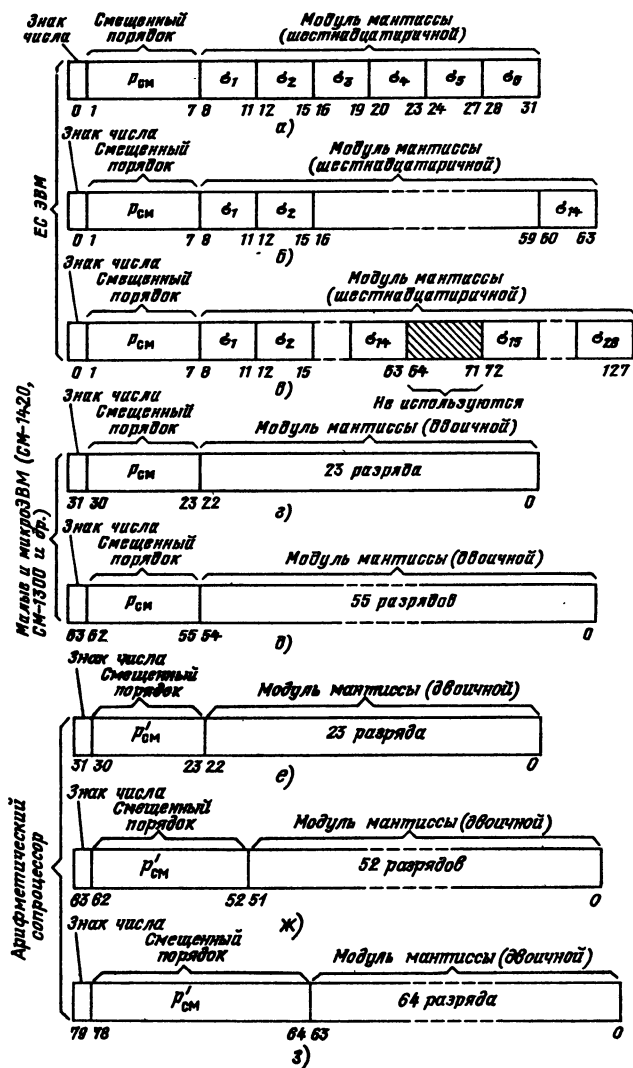


Рис. 2.2. Представление в ЭВМ чисел с плавающей точкой:  
а—в — шестнадцатеричные числа со смещенным порядком в форматах соответственно коротком, длинном и расширенном (ЕС ЭВМ); г, д — двоичные числа со смещенным порядком в форматах соответственно коротком и длинном (СМ-1420, СМ-1300); е—з — форматы двоичных чисел арифметического сопроцессора 8087

На рис. 2.2 показаны примеры форматов данных для чисел с плавающей точкой. Одна часть бит формата используется для представления порядка, а другая — мантиссы.

Особенности выполнения арифметических операций с числами с плавающей точкой рассматриваются в гл. 6. Арифметические действия над числами с плавающей точкой требуют выполнения помимо операций над мантиссами определенных операций над порядками (сравнение, вычитание и др.). Для упрощения операций над порядками их сводят к действиям над целыми положительными числами (целыми числами без знаков), применяя для представления порядков смещенный код (представление чисел с плавающей точкой со *смещенным порядком*).

В случае представления числа с плавающей точкой со смещенным порядком к его порядку  $p$  прибавляется целое число — смещение  $A=2^k$ , где  $k$  — число двоичных разрядов, используемых для модуля порядка.

Смещенный порядок  $p_{см} = p + A$  всегда положителен.

При фиксированном числе разрядов мантиссы любая величина представляется в машине с наибольшей возможной точностью нормализованным числом.

Число  $x = s^p q$  называется *нормализованным*, если мантисса  $q$  удовлетворяет условию

$$1 > |q| \geq 1/s. \quad (2.8)$$

В некоторых вычислительных устройствах используется другое условие нормализованности числа:

$$s > |q| \geq 1, \quad (2.8a)$$

т. е. старший разряд мантиссы в  $s$ -ричной системе отличен от нуля.

В процессе вычислений может получаться ненормализованное число. В этом случае машина, если это предписано командой, автоматически нормализует его («нормализация результата» операции).

Пусть  $r$  старших разрядов  $s$ -ричной мантиссы равно 0. Тогда нормализация заключается в сдвиге мантиссы на  $r$  разрядов влево и уменьшении порядка на  $r$  единиц, при этом в младшие  $r$  разрядов мантиссы записывается 0. После такой операции число не меняется, а условие (2.8) выполняется. При нулевой мантиссе нормализация невозможна.

В различных ЭВМ применяются представления чисел с плавающей точкой в системах счисления с различными основаниями, но равными целой степени числа 2 ( $s=2^m$ ), при этом по-

рядок  $p$  представляется целым числом, а мантисса  $q$  — числом, в котором группы по  $w$  двоичных разрядов изображают цифры мантиссы с основанием системы счисления  $s=2^w$ .

Примерами применяемых форм чисел с плавающей точкой с различными основаниями системы счисления являются

$$x=2^p q; \quad x=8^p q; \quad x=16^p q.$$

Использование для чисел с плавающей точкой недвоичного основания несколько уменьшает точность вычислений (при заданном числе разрядов мантиссы), но позволяет увеличить диапазон представляемых в машине чисел и ускорить выполнение некоторых операций, в частности нормализации, за счет того, что сдвиг может производиться сразу на несколько двоичных разрядов (на четыре разряда для  $s=16$ ). Кроме того, уменьшается вероятность появления ненормализованных чисел в ходе вычислений.

Диапазон представимых в машине чисел с плавающей точкой зависит от основания системы счисления и числа разрядов, выделенных для изображения порядка. Точность вычислений при плавающей точке определяется числом разрядов мантиссы. С увеличением числа разрядов мантиссы увеличивается точность вычислений, но увеличивается и время выполнения арифметических операций.

Задачи, решаемые на ЭВМ, предъявляют различные требования к точности вычислений. Поэтому во многих машинах используется несколько форматов с плавающей точкой с различным числом разрядов мантиссы.

В ЕС ЭВМ числа с плавающей точкой представляются в шестнадцатеричной (двоично-кодированной) системе счисления. На рис. 2.2, а—в показаны три используемых в ЕС ЭВМ формата с плавающей точкой: короткий (мантисса 6 шестнадцатеричных цифр), длинный (мантисса 14 шестнадцатеричных цифр) и расширенный (мантисса 28 шестнадцатеричных цифр). Во всех форматах крайний левый разряд является знаковым, а следующие семь разрядов служат для представления смещенного порядка при  $A=64$ . Смещенный порядок может принимать значения от  $p_{см}=127$  (при  $p=63$ ) до  $p_{см}=0$  (при  $p=-64$ ). Мантисса  $q$  рассматривается как число, составленное из шестнадцатеричных цифр в виде

$$q = \sum_{j=1}^k \sigma_j \cdot 16^{-j} \quad (\sigma_j = 0, 1, 2, \dots, F),$$

где  $k$  — число шестнадцатеричных разрядов мантиссы.

Шестнадцатеричное число с плавающей точкой считается нормализованным, если старшая шестнадцатеричная цифра  $\sigma_1$  отлична от 0. В двоично-кодированном представлении нормализованного шестнадцатеричного числа три старшие двоичные цифры могут равняться 0. Это несколько увеличивает относительную погрешность представления чисел при фиксированном числе разрядов мантииссы.

Сдвиг на один шестнадцатеричный разряд выполняется как сдвиг мантииссы сразу на четыре двоичных разряда. Для случая, изображенного на рис. 2.2, *а*, наибольшее число сдвигов, которое может потребоваться при нормализации, равно 5. Для чисел с плавающей запятой с двоичным основанием системы счисления в формате, представленном на рис. 2.2, *а*, наибольшее число сдвигов было бы равно 23.

В табл. 2.1 приведены примеры представления в ЕС ЭВМ чисел с плавающей точкой (в формате слова) в памяти машин. Следует обратить внимание на то, что при изображении кода в памяти шестнадцатеричными цифрами первые две старшие шестнадцатеричные цифры представляют совместно знак числа и смещенный порядок.

Представление чисел с плавающей точкой в малых и микро-ЭВМ (рис. 2.2, *г* и *д*) существенно отличается от принятого в ЕС ЭВМ. В этих машинах используется представление с двоичным основанием системы счисления. Имеются два формата: короткий и длинный, имеющие соответственно длину 32 и 64 разрядов. Для представления смещенного порядка отводится 8 разрядов (смещение  $A=128$ ). В коротком формате двоичная мантиисса имеет длину 23 разряда, а в длинном 55. Числа в памяти всегда представляются в нормализованной форме, при которой старший разряд мантииссы всегда равен 1. Поэтому старший разряд мантииссы не фиксируется, его единичное значение подразумевается. Таким образом, в этих машинах фактически обеспечивается точность представления чисел, соответствующая длинам мантиисс 24 и 56 разрядов. Использование подразумеваемого («скрытого») старшего разряда мантииссы приводит к необходимости представления чисел с нулевой мантииссой особым кодом, так как нулевая мантиисса неотличима от мантииссы, равной  $1/2$ . Таким кодом служит код, содержащий все нули в разрядной сетке.

Интересно отметить, что одинаковые по длине форматы чисел с плавающей точкой на рис. 2.2, *а* и *г*, использующие формы представления с 16-ричным и двоичным основаниями, имеют существенно различные диапазоны представимых чисел. В первом случае наибольшее представимое нормализованное число равно примерно  $10^{76}$ , а во втором — лишь  $10^{38}$ .



Т а б л и ц а 2.1. Представление чисел с плавающей точкой в ЕС ЭВМ

Десятичное число	Эквивалентное шестнадцатеричное число	Мантисса нормализованного шестнадцатеричного числа	Порядок нор- мализованного шестнадцате- ричного числа
$(16289)_{10}$	$(3FA1)_{16}$	$(0.3FA1)_{16}$	$(4)_{16}$
$(-16289)_{10}$	$(-3FA1)_{16}$	$(-0.3FA1)_{16}$	$(4)_{16}$
$(-0.01)_{10}$	$(-0.028F5C3)_{16}$	$(-0.28F5C3)_{16}$	$(-1)_{16}$
$(0.01)_{10}$	$(0.028F5C3)_{16}$	$(0.28F5C3)_{16}$	$(-1)_{16}$

Использование чисел с плавающей точкой в формате, изображенном на рис. 2.2, а, соответствует вычислениям примерно с семью десятичными разрядами, что для ряда научно-технических расчетов недостаточно из-за накопления ошибок округления. Поэтому в ЕС ЭВМ для чисел с плавающей точкой предусмотрены еще два изображенных на рис. 2.2, в и г формата: длинный и расширенный, занимающие соответственно два и четыре 32-разрядных слова. В этих форматах не меняется число разрядов для изображения порядка и, следовательно, сохраняется диапазон представляемых чисел, а длина мантиссы увеличивается соответственно до 14 и 28 шестнадцатеричных разрядов. Использование таких представлений чисел эквивалентно выполнению вычислений соответственно примерно с 16 и 32 десятичными разрядами.

Наличие в ЭВМ нескольких длин форматов для представления чисел позволяет с учетом требований задачи выбирать или короткие форматы для чисел и тем самым экономить емкость памяти, занимаемую данными, и снижать продолжительность отдельных операций ЭВМ, или использовать более длинные форматы для получения большей точности.

Для представления чисел с плавающей точкой разработан новый стандарт, который используется в новых разработках МП и микроЭВМ [35]. По-видимому, первым МП с представлением чисел с плавающей точкой согласно новому стандарту стал арифметический микропроцессор Intel 8087.

По этому стандарту для представления порядка используется смещенный код с отрицательным нулем (см. § 2.3), а условие нормализованности числа имеет вид  $2 > |q| \geq 1$ . Мантисса представляется прямым кодом (модуль и знак) со «скрытым» старшим разрядом, имеющим вес 1.

В 4- и 8-байтном словах формат чисел имеет вид, представленный на рис. 2.2, е, ж. Используется также и 10-байтное слово для представления результатов промежуточных вычислений (рис. 2.2, з).

Смещенный порядок	Представление чисел в памяти ЭВМ двоичным словом	Представление чисел в памяти ЭВМ шестнадцатеричным словом
$(44)_{16}$	0 1000100001111111010000100000000	443FA100
$(44)_{16}$	1 1000100001111111010000100000000	C43FA100
$(3F)_{16}$	1 0111111001010001111010111000011	BF28F5C3
$(3F)_{16}$	0 0111111001010001111010111000011	3F28F5C3

Согласно стандарту смещение порядка равно  $(2^{k-1} - 1)$ , где  $k$  — число разрядов кода порядка, а значение порядка лежит в интервале  $-(2^{k-1} - 1) \div -0,1 \div 2^{k-1} - 1$ . Код  $p_{см}$ , содержащий во всех разрядах 1, не используется. Он зарезервирован для указания на переполнение порядка или потерю значимости мантиссы ( $q=0$ ), при этом положительное и отрицательное переполнения идентифицируются соответственно положительной и отрицательной мантиссами, содержащими в обоих случаях 1 во всех цифровых разрядах. Указанием на потерю значимости мантиссы служит отрицательность мантиссы с 0 во всех цифровых разрядах. При нулевых кодах порядка и мантиссы представляемое число полагается равным нулю.

Рассматриваемые форматы обеспечивают чрезвычайно широкий диапазон представления чисел и одновременно небольшую погрешность. Так, для 8-байтного слова  $p'_{см}$  задает порядок  $p = -1023 \div +1023$ , что соответствует диапазону чисел примерно  $\pm 10^{-308} \div \pm 10^{308}$ , представляемых с погрешностью  $2^{-52}$ .

## 2.5. Кодирование десятичных чисел и алфавитно-цифровой информации

Современные ЭВМ обрабатывают не только числовую, но и текстовую, другими словами — алфавитно-цифровую информацию, содержащую цифры, буквы, знаки препинания, математические и другие символы. Именно такой характер имеют экономическая, планово-производственная, учетная информация, а также тексты программ на алгоритмических языках и другая информация. Характер этой информации такой, что для ее представления требуются слова переменной длины.

Возможность ввода, обработки и вывода алфавитно-цифровой информации важна и для решения чисто математических задач, так как позволяет оформлять результаты вычислений

в удобной форме — в виде таблиц с нужными заголовками и пояснениями.

Совокупность всех символов, используемых в вычислительной системе, представляет собой ее *алфавит*. Символу соответствует машинная единица информации *слог*. Так называют группу двоичных разрядов, служащую для представления символа в машине (двоичный код символа). Применяются различные варианты кодирования символов, использующие коды разной длины.

При выборе способа кодирования необходимо учитывать объем алфавита символов и требования, связанные с облегчением автоматической обработки данных.

Деловая информация в среднем содержит почти вдвое больше цифр, чем букв. Поэтому наряду с общей системой кодирования алфавитно-цифровых символов (десятичные цифры, буквы и другие знаки) в машинах сохраняют также отдельную систему кодирования для данных, состоящих только из десятичных цифр.

Необходимо, чтобы память машины эффективно использовала при размещении в ней как алфавитно-цифровой, так и десятичной информации.

Наибольшее распространение получило представление алфавитно-цифровой информации с помощью 8-разрядных слогов (рис. 2.3), называемых *байтами*. С помощью байта можно кодировать 256 различных символов.

Для представления алфавитно-цифровых символов в памяти машины и на носителях информации в ЕС ЭВМ и в некоторых других ЭВМ используется двоичный код для обработки информации (ДКОИ), а в микропроцессорах К1810, персональных компьютерах и в микроЭВМ — расширенный за счет букв русского алфавита код ASCII.

Алфавитно-цифровая информация представляется словами переменной длины, содержащими нужное число байт-символов (рис. 2.3). Например, в ЕС ЭВМ алфавитно-цифровое слово может иметь длину от 1 до 256 байт.

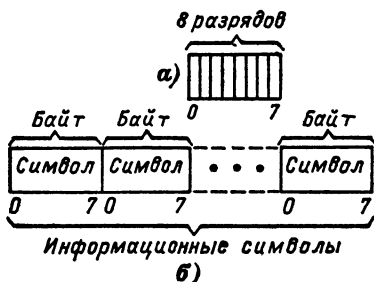


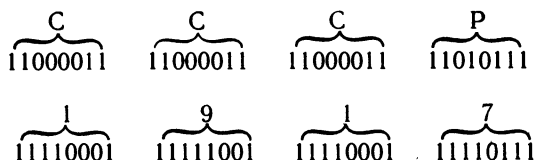
Рис. 2.3. Восемьразрядный слог (байт) (а) и размещение алфавитно-цифровых символов (б)

[illegible]

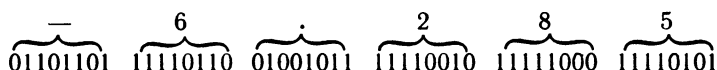
Для упрощения автоматизации обработки данных применяют весовой принцип кодирования символов. Двоичное число, соответствующее коду символа, называется его весом. При весовом кодировании веса кодов цифр последовательно возрастают, а веса кодов букв увеличиваются в алфавитном порядке. Вес кода буквы Б на 1 больше веса кода буквы А и т. д., а код пробела меньше веса кода буквы А.

Если необходимо расположить список фамилий в алфавитном порядке, то при весовом принципе кодирования эта операция может быть выполнена машиной путем сравнения двоичных чисел, соответствующих кодовым изображениям фамилий.

Рассмотрим примеры кодирования алфавитно-цифровой информации. При использовании кода ДКОИ (рис. 2.4) выражение «СССР 1917» будет храниться в памяти ЭВМ в виде алфавитно-цифрового слова из 8 байт:



а число — 6.285 запишется в память в виде слова из 6 байт:



Для экономии емкости памяти и для удобства выполнения арифметических операций над десятичными числами в машинах с байтовым представлением информации наряду с описанным выше способом представления алфавитно-цифровых символов предусматриваются специальные форматы для десятичных чисел — зонный («распакованный») (рис. 2.5, а) и уплотненный («упакованный») (рис. 2.5, б). Десятичные данные рассматриваются как десятичные числа со знаком. Числа могут иметь переменную длину. Для представления отрицательных чисел используется прямой код.

Зона	Цифра	Зона	Цифра		Зона	Цифра	Знак	Цифра
0	3/4	7	0	3/4	7	0	3/4	7
				а)				
Цифра	Цифра	Цифра	Цифра		Цифра	Цифра	Цифра	Знак
0	3/4	7	0	3/4	7	0	3/4	7
				б)				

Рис. 2.5. Представление десятичной информации (ЕС ЭВМ): а — зонный (распакованный) формат; б — упакованный формат

Десятичные цифры 0, 1, ..., 9 представляются в двоично-десятичной форме — коде 8421, в котором десятичная цифра изображается соответствующим 4-разрядным двоичным числом. Не используемые при этом комбинации 4-разрядных кодов (1010—1111) служат для кодирования знаков и служебных символов.

Код 8421 удобен для выполнения машиной (а не вручную) преобразований из десятичной системы в двоичную и обратно. Этот код аддитивен, т. е. сумма представлений двух цифр есть код их суммы. Однако использование этого кода связано с трудностями обнаружения переноса в следующий десятичный разряд и сложностью перехода к обратным и дополнительным кодам для десятичных чисел, облегчающим выполнение алгебраического сложения. Это объясняется тем, что код 8421 не является самодополняющимся, т. е. инверсия его двоичных цифр не дает кода дополнения десятичной цифры до 9.

Для зонного формата в каждом байте содержатся только одна десятичная цифра и служебный знак (зона), при этом четыре правых разряда байта служат для представления десятичной цифры в двоично-десятичном коде, а четыре левых заняты специальным 4-разрядным кодом, называемым *зоной*. Младший байт в этом формате состоит из кодов знака и младшей десятичной цифры числа.

В ДКОИ принято кодировать: плюс — 1100, минус — 1101 и зона — 1111.

Число — 6.285 в зонном десятичном формате (при работе с кодом ДКОИ) будет иметь вид

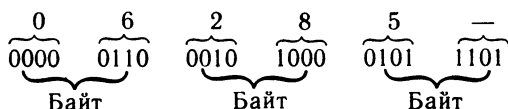
$$\begin{array}{ccccccc} & 6 & & 2 & & 8 & & - & 5 \\ \hline 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & & 1 & 1 & 1 & 1 & 0 & 0 & 0 & & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}$$

Информация при этом о месте расположения десятичной запятой (точки) фиксируется в самой программе задачи.

При кодировании десятичного числа без знака в левые четыре разряда младшего байта записывается код зоны.

Важное достоинство 8-разрядного слова — байта состоит в возможности размещения в одном слове кодов двух десятичных цифр, чем достигается существенная экономия памяти. Эта возможность реализуется в упакованном десятичном формате (рис. 2.5, б). В этом формате код знака размещается в правых четырех разрядах младшего байта. Десятичное число всегда занимает целое число байтов. Если левые четыре разряда самого левого (старшего) байта оказываются свободными, они заполняются нулями.

Число — 6.285 в упакованном десятичном формате имеет вид



В ЕС ЭВМ в упакованном формате десятичное число может иметь длину от 1 до 16 байт. В этом случае максимальная длина числа — это 31 десятичная цифра плюс знак. При выполнении операций над десятичными числами в ЕС ЭВМ используется упакованный формат. Результат получается также в этом формате. Числа, участвующие в операции, могут иметь неодинаковую длину. Они рассматриваются как целые числа, выравненные по младшим разрядам. Формат с зоной используется при операциях ввод-вывода десятичных данных.

В вычислительных машинах, использующих форматы, показанные на рис. 2.5, имеются команды для преобразования десятичных чисел из зонного формата в упакованный и обратно.

### Контрольные вопросы

1. Перевести шестнадцатеричное число A51.OE4 в восьмеричную форму.

2. Разделить двоичное число 10100010 на 1001.

3. Представить десятичное число — 115 двоичным 8-разрядным прямым, обратным и дополнительным кодами.

4. Определить, какие десятичные числа задают слова C1A513FF и 3E1D8000, если это:

число без знака ЕС ЭВМ;

число со знаком ЕС ЭВМ;

число с плавающей точкой ЕС ЭВМ;

число с плавающей точкой СМ-1300.

5. Сопоставить диапазоны представимых в машинах ЕС ЭВМ и СМ ЭВМ чисел с плавающей точкой (см. рис. 2.2).

6. Записать свою фамилию и год рождения в коде ДКОИ.

## Глава 3

# ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ И ТИПОВЫЕ УЗЛЫ

## 3.1. Представление информации физическими сигналами

В предыдущей главе было показано, как информация представляется в двоичном алфавите. Физическими аналогами знаков 0 и 1 двоичного алфавита служат сигналы, способные при-

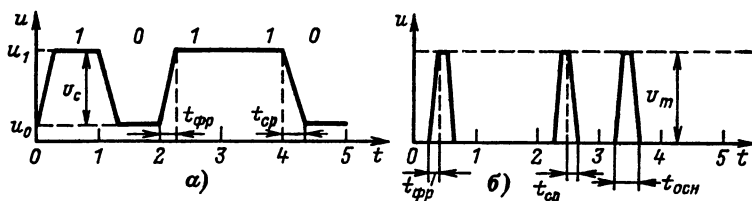


Рис. 3.1. Способы представления цифровой информации:  
 а — сигналы потенциального типа; б — сигналы импульсного типа

нимать два хорошо различимых значения, например напряжение (потенциал) высокого и низкого уровней, отсутствие и наличие электрического импульса, противоположные по знаку значения магнитной индукции и т. п.

В схемах цифровых устройств переменные и соответствующие им сигналы изменяются не непрерывно, а лишь в дискретные моменты времени, обозначаемые целыми неотрицательными числами: 0, 1, 2, ...,  $i$ ...

Временной интервал между двумя соседними моментами дискретного времени называется *тактом*.

Во многих случаях цифровые устройства содержат специальный блок, вырабатывающий синхронизирующие сигналы (СС), отмечающие моменты дискретного времени (границы тактов).

В цифровых вычислительных устройствах обычно применяют потенциальный и импульсный способы физического представления информации.

При потенциальном способе (рис. 3.1, а) двум значениям переменной 1 и 0 соответствуют разные уровни напряжения в соответствующей точке схемы машины (*потенциальный код*). Потенциальный сигнал сохраняет постоянный уровень в течение такта, а его значение в переходные моменты не является определенным.

При импульсном способе представления информации (рис. 3.1, б) единичное и нулевое значения двоичной переменной отображаются наличием и отсутствием электрического импульса или разнополярными импульсами в соответствующей точке схемы (*импульсный код*).

Импульсный сигнал можно характеризовать амплитудой  $U_m$ , продолжительностью импульса по основанию  $t_{осн}$ , длительностью фронта  $t_{фр}$  и среза  $t_{ср}$  (рис. 3.1, б).

Аналогичные понятия могут быть применены и к потенциальному сигналу (рис. 3.1, а). Потенциальный сигнал характеризуется, кроме того, разностью  $U_c$  верхнего и нижнего уровней напряжения. Понятия фронта и среза у потенциального сигнала



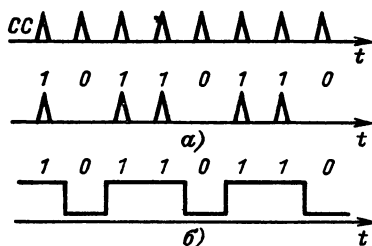


Рис. 3.2. Последовательный импульсный код (а) и последовательный потенциальный код (б)

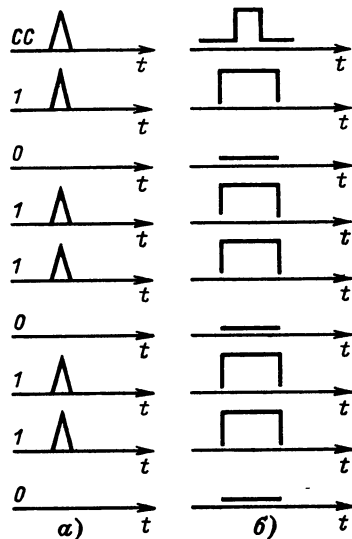


Рис. 3.3. Параллельный импульсный код (а) и параллельный потенциальный код (б)

всегда связаны с процессом перехода соответственно от нижнего к верхнему и от верхнего к нижнему уровню напряжений.

В соответствии с типом используемых сигналов для представления информации схемы цифровых устройств принято делить на импульсные, потенциальные и импульсно-потенциальные.

Слово может быть представлено последовательным или параллельным кодом.

При *последовательном коде* каждый временной такт предназначен для отображения одного разряда кода слова (рис. 3.2). В этом случае все разряды слова фиксируются по очереди одним и тем же элементом и проходят через одну линию передачи информации.

При *параллельном коде* все разряды двоичного кода слова представляются в одном временном такте, фиксируются отдельными элементами и проходят через отдельные линии, каждая из которых служит для представления и передачи только одного разряда слова.

При параллельной передаче информации код слова развертывается не во времени, а в пространстве, так как значения всех разрядов слова передаются по нескольким линиям одновременно (рис. 3.3).

В зависимости от применяемого кода устройства вычислительной техники называются последовательными или параллельными. При использовании последовательного кода все операции, в том числе передача слов из одного узла в другой, производятся

поочередно для каждого разряда слова, и поэтому последовательные устройства работают медленнее, чем параллельные. В современных ЭВМ основные устройства, участвующие в обработке информации, для достижения высокого быстродействия строятся как параллельные, хотя они и требуют большего объема аппаратуры. Для экономии оборудования в некоторых устройствах применяют последовательно-параллельный код, при котором слова разбиваются на части (слоги) и передача, а иногда и обработка производятся последовательно слог за слогом, при этом каждый слог представляется параллельным кодом.

### **3.2. Понятие о комбинационной схеме и цифровом автомате**

Устройство, преобразующее дискретную информацию, в общем случае имеет  $n$  входов для входных сигналов и  $m$  выходов, с которых снимаются выходные сигналы.

Каждый из входных сигналов соответствует некоторому символу (букве) входного алфавита. В свою очередь, выходные сигналы представляют собой символы (буквы) выходного алфавита. В качестве букв этих алфавитов обычно используются двоичные и, реже, десятичные цифры.

Преобразование информации в ЭВМ производится электронными устройствами (логическими схемами) двух классов: *комбинационными схемами* и *цифровыми автоматами*.

В комбинационных схемах (КС) совокупность выходных сигналов (выходное слово  $Y$ ) в любой момент времени однозначно определяется входными сигналами (входным словом  $X$ ), поступающими на входы в тот же момент времени (рис. 3.4, а).

Реализуемый в этих схемах способ обработки информации называется комбинационным, так как результат обработки информации зависит только от комбинации входных сигналов и вырабатывается сразу при подаче входной информации.

Закон функционирования КС определен, если задано соответствие между ее входными и выходными словами, например, в виде таблицы.

Это соответствие может быть задано и в аналитической форме с использованием булевых функций.

Другой, более сложный класс преобразователей дискретной информации составляют цифровые автоматы. Цифровой автомат в отличие от комбинационной схемы имеет некоторое конечное число различных внутренних состояний.

Под воздействием входного слова цифровой автомат переходит из одного состояния в другое и выдает выходное слово. Выходное слово на выходе цифрового автомата в такте опреде-

ляется в общем случае входным словом, поступившим в этот такт на вход автомата, и внутренним состоянием автомата, которое явилось результатом воздействия на автомат входных слов в предыдущие такты.

Комбинация входного слова и текущего состояния автомата в данном такте определяет не только выходное слово, но и то состояние, в которое автомат перейдет к началу следующего такта.

Цифровой автомат содержит память, состоящую из запоминающих элементов (ЗЭ) — триггеров, элементов задержки и других элементов, фиксирующих состояние, в котором он находится. Комбинационная схема не содержит ЗЭ. Поэтому ее называют автоматом без памяти или примитивным автоматом.

Структурная схема цифрового автомата, представленная на рис. 3.4, б, содержит запоминающие элементы  $ЗЭ_1 — ЗЭ_k$  и комбинационные схемы  $КС_I$  и  $КС_{II}$ .

Состояния ЗЭ, определяющие состояние автомата, передаются в форме сигналов  $q_i$  по цепям прямой связи на входы  $КС_{II}$  и по цепям обратной связи на входы  $КС_I$ . На входы комбинационных схем поступают также сигналы  $x_1, \dots, x_n$  с входов автомата.

Выходное слово вырабатывается в  $КС_{II}$ , причем входными переменными для нее служат буквы входного слова и состояния ЗЭ — состояние автомата. Выходные сигналы  $КС_I$  переводят автомат (его ЗЭ) в новое состояние, при этом входными переменными для этой схемы служат буквы входного слова и состояния ЗЭ. Одновременность появления новых значений входных сигналов на всех входах устройств достигается с помощью тактирующих сигналов, называемых также синхросигналами и обеспечивающих передачу информации с ЗЭ на входы комбинационной схемы одновременно с сигналами, поступающими на ее входы с других устройств.

В ряде случаев при анализе автомата его заменяют автоматом с одним эквивалентным входом и одним эквивалентным выходом и считают, что эквивалентные входной сигнал  $x(t)$  и выходной сигнал  $y(t)$  принимают значения из соответствующим образом преобразованных алфавитов  $P$  и  $S$  входных и выходных сигналов.

Для задания цифрового автомата должны быть указаны:

- 1) входной алфавит  $P = \{p_1, p_2, \dots, p_N\}$ ;
- 2) выходной алфавит  $S = \{s_1, s_2, \dots, s_M\}$ ;
- 3) алфавит состояний  $Q = \{Q_0, Q_1, \dots, Q_r\}$ ;
- 4) начальное состояние автомата  $Q_0$ ;

5) функции переходов  $A(Q, x)$  и выходов  $B(Q, x)$ , однозначно определяющие зависимость соответственно состояния

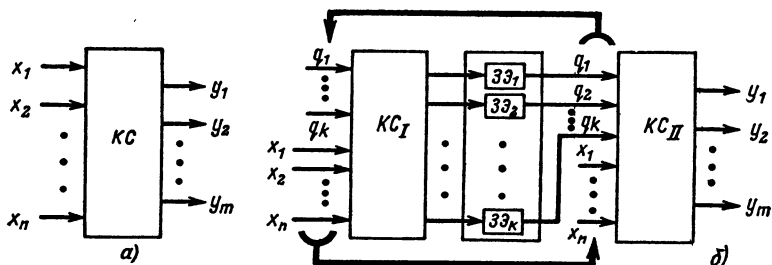


Рис. 3.4. Комбинационная схема (а) и цифровой автомат (б)

автомата  $Q(t+1)$  в такте  $(t+1)$  и выходного сигнала  $y(t)$  от состояния автомата  $Q(t)$  и входного сигнала  $x(t)$  в такте  $t$ .

Используя функции переходов и выходов, можно поведение автомата описать выражениями

$$Q(t+1) = A[Q(t), x(t)]; \quad (3.1)$$

$$y(t) = B[Q(t), x(t)], \quad (3.2)$$

где  $t=0, 1, 2 \dots$ ;  $Q(0) = Q_0$ .

Выражениям (3.1) и (3.2) соответствует автомат, выходной сигнал которого зависит от состояния автомата и сигнала на его входе. Такой автомат называется *автоматом Мили*.

В устройствах ЭВМ широко применяются и так называемые *автоматы Мура*, у которых выходной сигнал  $y(t)$  в такте  $t$  зависит исключительно от состояния автомата  $Q(t)$  в этом такте и не зависит от входного сигнала  $x(t)$ .

Функционирование автомата Мура описывается выражениями

$$Q(t+1) = A[Q(t), x(t)]; \quad (3.3)$$

$$y(t) = B[Q(t)], \quad (3.4)$$

где  $t=0, 1, 2 \dots$ ;  $Q(0) = Q_0$ .

Функции переходов и выходов могут задаваться различными способами, например в форме таблиц или с помощью графов. При задании в виде графов (см. гл. 8) состояния автомата представляются вершинами, а переходы из состояния в состояние — дугами. На дугах указываются значения входных сигналов, вызывающих соответствующие переходы. Выходные сигналы автомата Мура указываются рядом с вершинами графа. Входные сигналы автомата Мили, вырабатываемые перед переходом, указываются на соответствующих дугах.

В теории автоматов вводятся понятия *полной системы переходов* и *полной системы выходов автомата*. Если для двух любых состояний  $Q_i$  и  $Q_j$  автомата имеется входной сигнал, переводящий автомат из состояния  $Q_i$  в  $Q_j$ , то такой автомат называется автоматом с полной системой переходов. Автомат Мура имеет полную систему выходов, если выходные сигналы различны для всех его состояний.

При построении узлов ЭВМ, являющихся цифровыми автоматами, в качестве запоминающих элементов (элементов памяти) используются элементарные автоматы. Элементарными автоматами служат автоматы Мура с двумя состояниями, обладающие полными системами переходов и выходов.

Для описания законов функционирования комбинационных схем используется математический аппарат булевых функций. Напомним, что переменные  $x_1, x_2, \dots, x_n$  называются двоичными, если они могут принимать только два значения: 0 и 1, а функция от двоичных переменных  $f(x_1, x_2, \dots, x_n)$  называется *булевой*, если она, так же как и ее аргументы, принимает только два значения: 0 и 1.

Система булевых функций  $W$  называется *функционально полной*, если для любой булевой функции  $f(x_1, \dots, x_n)$  может быть построена равная ей функция путем суперпозиции функций  $x_1, \dots, x_n$  и функций системы  $W$ , взятых в любом конечном числе экземпляров каждая.

В математической логике доказывается, что если система булевых функций содержит функции  $x_1x_2$  (конъюнкция),  $x_1 \vee x_2$  (дизъюнкция),  $\bar{x}$  (инверсия), то она является функционально полной.

Из этой системы можно исключить некоторые функции без нарушения функциональной полноты. Действительно, функциональной полнотой будут обладать также система булевых функций  $x_1 \vee x_2, \bar{x}$  и система  $x_1x_2, \bar{x}$ .

Функционально полной является система, состоящая из одной булевой функции И — НЕ («штрих Шеффера»):  $\overline{x_1x_2}$ ; функциональной полнотой обладает и система, содержащая единственную булеву функцию ИЛИ — НЕ («стрелка Пирса»):  $\overline{x_1 \vee x_2}$ . Существуют и другие функционально полные системы булевых функций.

Техническим аналогом булевой функции является комбинационная схема, выполняющая соответствующее этой функции преобразование информации.

Постоянные уровни напряжения, соответствующие принятому в схеме представлению сигналов 0 и 1, могут рассматриваться как технические аналоги функции констант 0 и 1.

Логические операции над двоичными переменными реализуются схемами, которые называются *комбинационными логическими элементами*. Число входов комбинационного логического элемента соответствует числу аргументов воспроизводимых им одной или нескольких булевых функций.

Подобно тому как сложная булева функция может быть получена суперпозицией более простых функций, так и комбинационная схема строится из элементарных схем — из комбинационных логических элементов, при этом легко установить технический аналог операции суперпозиции. Последовательное соединение комбинационных схем, в том числе комбинационных логических элементов, при котором выход одной схемы соединен со входом другой, соответствует подстановке в булевы функции в качестве аргументов других булевых функций. Пересоединение на входах комбинационных схем соответствует перестановке аргументов булевых функций.

Логические элементы могут быть не только комбинационными. Используются также логические элементы, реализующие функции элементарных автоматов. Подробнее о логических элементах см. в § 3.3 и 3.4.

Схему, показывающую связи между различными логическими элементами, где сами элементы представлены условными обозначениями, называют *функциональной*.

При вычерчивании функциональных схем логические элементы, а часто и более крупные части схем изображают прямоугольниками, в которые вписывают символы реализуемых функций, при этом если функция реализуется при инверсных значениях некоторых входных переменных, то соответствующие входы отмечаются кружком. Выход отмечается кружком, если функция реализуется с инверсией. Примеры схем, реализующих булевы функции, даны на рис. 3.5.

Набор логических элементов для построения комбинационных схем является *функционально полным*, если реализуемые этими элементами булевы функции образуют функционально полную систему функций.

Комплекс элементов обладает *функциональной полнотой для построения цифровых автоматов*, если он содержит функционально полный набор логических элементов для построения комбинационных схем и элементарный автомат с полной системой выходов и переходов.

В ЭВМ в качестве элементарных автоматов используются главным образом триггеры. Описание триггеров различных типов приведено в § 3.4.

Особенности построения цифровых устройств связаны со способом передачи информации между логическими элементами.

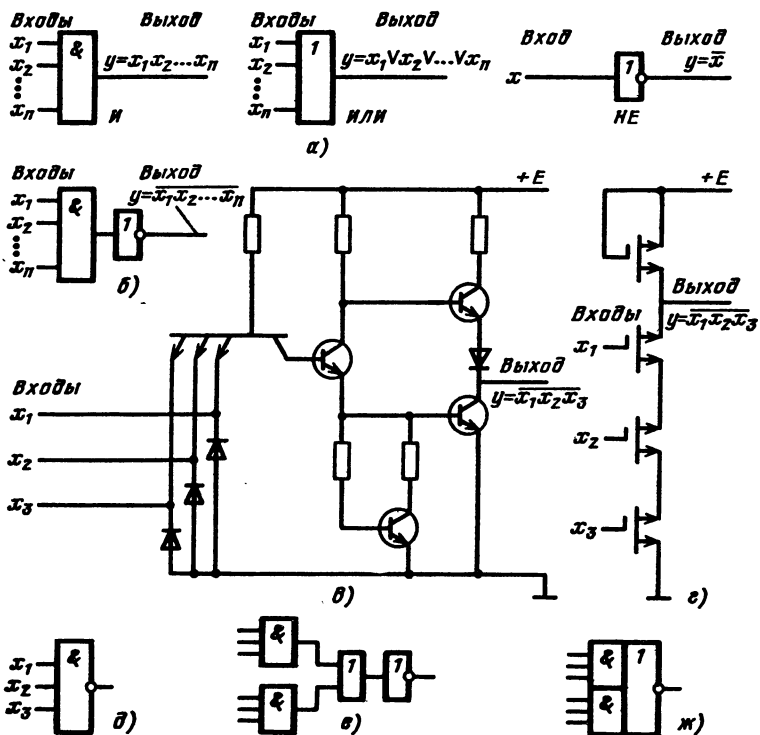


Рис. 3.5. Схемы логических элементов:

а — элементы И, ИЛИ, НЕ; б — функциональная схема элемента И — НЕ; в — принципиальная электрическая схема И — НЕ транзисторно-транзисторной логики; г — принципиальная электрическая схема элемента И — НЕ на МОП-транзисторах; д — условное обозначение элемента И — НЕ; е — функциональная схема элемента И — ИЛИ — НЕ; ж — условное обозначение элемента И — ИЛИ — НЕ

Передача информации от элемента к элементу может осуществляться несинхронизируемым (асинхронным) и синхронизируемым способами, причем последний используется только при передаче информации в запоминающие элементы.

При несинхронизируемом способе передачи информации входные сигналы логических элементов преобразуются с небольшой задержкой в выходные сигналы, которые непосредственно воздействуют на входы следующих логических элементов.

При синхронизируемом способе передачи информации входные сигналы воздействуют на запоминающие логические элементы в строго определенные моменты времени, соответствующие появлению синхронизирующих сигналов.

Передача информации между запоминающими логическими элементами в общем случае требует выполнения следующего условия: она производится только после завершения передачи информации о предыдущем состоянии принимающего информацию элемента другому ЗЭ. Для выполнения этого условия в потенциальных схемах используется несколько серий синхронизирующих сигналов, сдвинутых во время относительно друг друга так, что когда сигнал одной серии принимает значение 1, то сигнал другой серии принимает значение 0.

Последовательное соединение схем логических элементов, позволяющее выполнять передачу информации, естественно, подразумевает возможность соединения выходов одних элементов со входами других. Соединение между собой выходов элементов в общем случае считается недопустимым. Однако существуют элементы, выходы которых предназначены именно для работы в условиях взаимного соединения. Такие элементы позволяют строить схемы обработки информации, в которых несколько элементов — источников сигналов — передают информацию приемнику (или нескольким приемникам) по одной общей соединительной линии, к которой присоединены выходы нескольких источников. Схема каждого источника строится таким образом, чтобы его выход мог находиться в электрически отключенном состоянии от соединительной линии либо в состоянии 0 или 1 при подключении к соединительной линии. В каждый данный момент времени к соединительной линии подключается не более одного источника, отключенные электрически от линии источники не влияют на работу друг друга и подключенного к линии источника. Рассматриваемые элементы обычно называются *элементами с отключаемыми выходами (выходами с тремя состояниями)*.

Следует отметить, что некоторые логические элементы допускают объединение выходов как средство формирования более крупных элементов из исходных. В этом случае соединение выходов приводит к реализации некоторой «монтажной» булевой функции над значениями соединяемых выходов; обычно это «монтажное И» либо «монтажное ИЛИ». Допустимость «монтажной логики» обуславливается особенностями реализации элементов и при возможности оговаривается в технической документации.

### 3.3. Системы логических элементов

Системой (комплексом или серией) логических элементов ЭВМ называется предназначенный для построения цифровых устройств функционально полный набор логических элементов,



объединяемых общими электрическими, конструктивными и технологическими параметрами и использующих одинаковый способ представления информации и одинаковый тип межэлементных связей. Система элементов чаще всего избыточна по своему функциональному составу, что позволяет строить схемы, более экономные по числу использованных элементов. Системы элементов содержат элементы для выполнения логических операций, запоминающие элементы, реализующие функции узлов ЭВМ, а также элементы для усиления, восстановления и формирования сигналов стандартной формы.

Элементы представляют собой микроминиатюризованные интегральные электронные схемы (микросхемы), сформированные в полупроводниковом кристалле с помощью специальных технологических процессов.

Системы (серии) элементов могут содержать различные по сложности микросхемы: малой степени интеграции (ИС), средней степени интеграции (СИС) и большой степени интеграции (БИС). Логические элементы в виде ИС реализуют совокупность логических операций, таких, как И, ИЛИ, И — ИЛИ, И — НЕ, ИЛИ — НЕ, И — ИЛИ — НЕ и триггеры. Логические элементы на СИС и БИС реализуют узлы ЭВМ.

Основными параметрами системы логических элементов являются уровни питающих напряжений и сигналов для представления логических 0 и 1, нагрузочная способность (коэффициент разветвления по выходу), помехоустойчивость, рассеиваемая мощность, быстродействие.

Основные, наиболее часто употребляемые типы интегральных элементов — это потенциальные элементы транзисторно-транзисторной логики (ТТЛ), потенциальные элементы транзисторной логики с эмиттерными связями (ЭСЛ) и элементы на МОП-транзисторах.

Схемы ТТЛ относятся к интегральным логическим элементам среднего быстродействия. Время задержки сигнала в элементе ТТЛ составляет 5—10 нс. Нагрузочная способность схем ТТЛ допускает подключение к выходу элемента до десяти логических схем.

Более высоким быстродействием обладают интегральные микросхемы с эмиттерными связями (ЭСЛ), в которых транзисторы не входят в насыщение. Элементы ЭСЛ работают по принципу переключения токов при малых изменениях входных напряжений. Вследствие этого элементы ЭСЛ часто называют схемами с переключателями тока.

Время задержки элемента ЭСЛ меньше, чем элемента ТТЛ, и обычно имеет значение 1—2 нс.

Интегральные микросхемы на МОП (металл — окисел —

полупроводник)-транзисторах являются более медленно действующими, чем элементы ТТЛ или ЭСЛ. Время задержки элемента на МОП-транзисторах обычно 50—100 нс. Однако эти элементы отличаются меньшей потребляемой мощностью, большой нагрузочной способностью и помехоустойчивостью и, что особенно важно, требуют меньшей площади на поверхности интегральной микросхемы. Схемы на МОП-транзисторах технологичны и дешевы. Поэтому они находят широкое применение, особенно в цифровых устройствах, не требующих очень высокого быстродействия, или в устройствах, для которых важна очень высокая степень интеграции, в таких, как, например, устройства памяти (см. гл. 4).

МОП-транзисторы бывают  $n$ - и  $p$ -типов. Строят схемы и с одновременным использованием транзисторов  $n$ - и  $p$ -типов (дополняющие транзисторы).

Схемы с дополняющими транзисторами (К-МОП-схемы) отличаются малой потребляемой мощностью и более высоким быстродействием (10—50 нс), так как в цепях заряда и разряда паразитных емкостей схемы оказываются включенными малые сопротивления открытых транзисторов.

### 3.4. Триггеры

Триггер является элементом, который может находиться в одном из двух устойчивых состояний. Одному из этих состояний приписывается значение 1, а другому 0. Состояние триггера распознается по его выходному сигналу. Под влиянием входного сигнала триггер может скачкообразно переходить из одного устойчивого состояния в другое, при этом скачкообразно изменяется уровень напряжения его выходного сигнала.

Для удобства использования в схемах вычислительных устройств триггеры обычно имеют два выхода: прямой  $Q$  (называется также «выход 1») и инверсный  $\bar{Q}$  («выход 0»). В единичном состоянии триггера на выходе  $Q$  высокий уровень сигнала, а в нулевом — низкий. На выходе  $\bar{Q}$  наоборот.

Схемы триггеров можно разделить на несколько типов: с установочными входами —  $RS$ -триггер, со счетным входом —  $T$ -триггер, а также  $D$ -триггер,  $JK$ -триггер и др.

Если хотя бы по одному входу информация в триггер заносится принудительно под воздействием синхронизирующего сигнала, то триггер называется синхронизируемым (синхронным). Если занесение информации в триггер по любому входу производится без синхронизирующего сигнала, то триггер называется несинхронизируемым (асинхронным).

Общая форма условного обозначения триггеров показана на

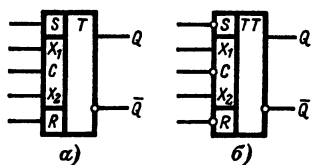


Рис. 3.6. Условные обозначения триггеров:

**а** — синхронизируемый одноктактный триггер с дополнительными входами **S** и **R** несинхронизируемой установки; **б** — синхронизируемый двухтактный триггер с дополнительными входами **S** и **R** несинхронизируемой установки

рис. 3.6. В основном поле ставится символ **T** или **TT** для обозначения соответственно одноктактного или двухтактного триггера. Дополнительное поле может быть разделено на две части: асинхронную и синхронную. В первой проставляются символы **R** и **S** входов несинхронизируемой установки триггера в 1 и 0, во второй — на местах **X<sub>1</sub>** и **X<sub>2</sub>** символы в соответствии с типом триггера, при этом используют следующие обозначения для входов:

- S** — вход установки триггера в 1;
- R** — вход установки триггера в 0;
- T** — вход триггера со счетным входом;
- D** — вход **D**-триггера;
- J** — вход для синхронизируемой установки состояния 1 в **JK**-триггере;
- K** — вход для синхронизируемой установки состояния 0 в **JK**-триггере;
- C** — вход синхронизации.

Если вход отмечен кружком, это означает, что действующее значение входного сигнала — 0.

Например, отсутствие кружка на входе **C** на рис. 3.6, **а** указывает на то, что входная информация заносится в триггер при единичном значении синхронизирующего сигнала (действующее значение синхронизирующего сигнала равно 1); кружок на входе **C** (рис. 3.6, **б**) означает, что прием информации происходит при нулевом значении синхронизирующего сигнала (действующее значение синхронизирующего сигнала равно 0).

Состояние триггера определяется сигналом **Q** на прямом выходе триггера (или сигналом **Q̄** на его инверсном выходе).

Законы функционирования триггеров задаются таблицами переходов с компактной записью, при которой в столбце состояний может быть указано, что новое состояние совпадает с предыдущим либо является его отрицанием.

**Асинхронный RS-триггер.** Асинхронный (несинхронизируемый) **RS**-триггер на интегральных элементах ИЛИ — НЕ показан на рис. 3.7. Триггер образован из двух комбинационных схем ИЛИ — НЕ, соединенных таким образом, что возникают положительные обратные связи, благодаря которым в устойчивом состоянии выходной транзистор одной схемы ИЛИ — НЕ закрыт, а другой открыт. Таблица 3.1, называемая таблицей переходов, определяет закон функционирования этого триггера.

Таблица 3.1. Таблица переходов RS-триггера

R	S	Q	Примечание
0	0	Q	Хранение
0	1	1	Установка 1
1	0	0	Установка 0
1	1	—	Запрещено

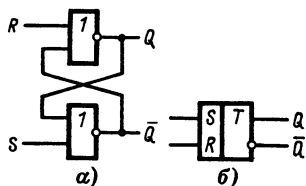


Рис. 3.7. Асинхронный RS-триггер на элементах ИЛИ — НЕ:

а — функциональная схема;  
б — условное обозначение

При  $R=1$  и  $S=0$  триггер устанавливается в нулевое состояние ( $Q=0$ ); при  $R=0$  и  $S=1$  он устанавливается в единичное состояние ( $Q=1$ ); при  $R=S=0$  триггер сохраняет состояние, в котором он находился до момента поступления на его входы нулевых сигналов. При  $R=S=1$  на прямом и инверсном выходах устанавливается нулевой сигнал. Триггерное кольцо превращается в два независимых инвертора, и при переходе к хранению ( $R=S=0$ ) триггер может устанавливаться в любое состояние. Поэтому такая комбинация входных сигналов обычно не используется.

Функционирование RS-триггера можно описать выражением

$$Q(t+1) = S(t) \vee Q(t) \bar{R}(t), \quad (3.5)$$

причем  $S(t)R(t)=0$ , и  $t$  — момент времени, предшествующий смене состояния.

**Синхронизируемый одноктактный RS-триггер.** На рис. 3.8 приведена схема синхронизируемого одноктактного RS-триггера на элементах И — НЕ. Здесь элементы 1 и 2 образуют схему входной логики асинхронного RS-триггера, построенного на элементах 3 и 4. Такие RS-триггеры имеют два информационных входа  $R$  и  $S$  и вход синхронизации  $C$ . Кроме того, триггер может иметь несинхронизируемые входы  $R$  и  $S$ . В этом случае функционирование триггера осуществляется либо под воздействием несинхронизируемых входов при  $C=0$ , либо под воздействием синхронизируемых входов. В последнем случае на несинхронизируемых входах должны присутствовать сигналы, которые не влияют на состояние схемы.

Переходы RS-триггера, построенного на элементах И — НЕ для синхронизируемых входов  $R$  и  $S$ , могут быть описаны табл. 3.1. Работа в соответствии с данной таблицей осуществляется при сигнале несинхронизируемого входа  $R=1$  и при  $C=1$ .

Входная информация, представленная в парафазном коде, заносится в синхронизируемый одноктактный RS-триггер через элементы входной

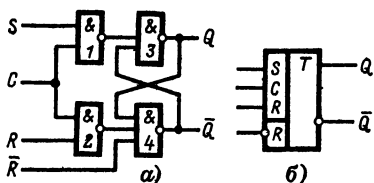


Рис. 3.8. Синхронизируемый одноктактный RS-триггер на элементах И — НЕ:

а — функциональная схема;  
б — условное обозначение

логики 1 и 2 в момент поступления сигнала синхронизации  $C$ . В отсутствие сигнала синхронизации триггер может быть установлен в состояние 0 путем подачи на несинхронизируемый вход  $R$  сигнала  $R=0$ .

**Двухтактный RS-триггер.** Устойчивая работа одноктактных RS-триггеров в схеме с передачей информации между триггерами возможна только в случае, если занесение в триггер информации осуществляется после завершения передачи информации о прежнем его состоянии в другой триггер. Это достаточно просто обеспечивается при использовании двух серий находящихся в противофазе синхросигналов.

Такой принцип обмена информацией реализован в двухтактных RS-триггерах.

Простейшая схема двухвходового двухтактного RS-триггера показана на рис. 3.9. Она состоит из двух одноктактных RS-триггеров и инвертора в цепи синхронизации. При поступлении на вход RS-триггера сигнала  $C=1$  входная информация заносится в первый одноктактный RS-триггер, а второй при этом будет хранить информацию, относящуюся к предыдущему периоду представления. По окончании действия сигнала синхронизации, когда  $C=0$ , а  $\bar{C}=1$ , первый RS-триггер перейдет в режим хранения, а второй примет то же состояние, что и первый. В результате к следующему такту на выходе двухтактного RS-триггера появится сигнал нового состояния. Таблица 3.2 задает закон функционирования такого двухтактного триггера. Такты  $t$  здесь задаются интервалами времени, в которые  $C=1$ . Двухтактный триггер изменяет свои состояния только после окончания действия сигнала синхронизации  $C=1$  (переход в режим хранения информации). Поэтому из двухтактных триггеров можно строить произвольные схемы, в том числе подавать сигналы с выхода триггера на его вход.

Для установки триггера в состояние 0 или 1 без использования синхросигналов в схему вводят дополнительные входы  $\bar{R}$  и  $\bar{S}$  несинхронизируемой установки. Связи с этими входами показаны на рис. 3.9, *а* штриховыми линиями. При подаче 0 на вход  $\bar{R}$  ( $\bar{S}$ ) и 1 на вход  $\bar{S}$  ( $\bar{R}$ ) оба одноктактных триггера устанавливаются в состояние 0 (1). При подаче 1 на оба этих входа работа триггера осуществляется в соответствии с табл. 3.2.

Схема RS-триггера составляет основу для построения других триггерных схем, таких, как T-, D- и JK-триггеры.

**T-триггер.** Триггер со счетным входом (T-триггер) в простейшем случае может быть построен с использованием двухтактного синхронизи-

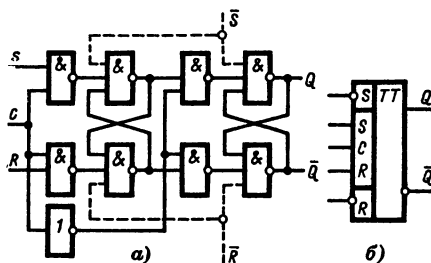


Рис. 3.9. Двухтактный RS-триггер на элементах И — НЕ:  
а — функциональная схема; б — условное обозначение

Т а б л и ц а 3.2. Таблица переходов двухтактного RS-триггера на элементах И—НЕ

$t$		$t+1$	Примечание
$R$	$S$	$Q$	
0	0	$Q(t)$	Хранение
1	0	0	Установка 0
0	1	1	Установка 1
1	1	—	Запрещено

руемого RS-триггера. T-триггер должен реализовать функцию вида

$$Q(t+1) = Q(t) \bar{T}(t) \vee \bar{Q}(t) T(t). \quad (3.6)$$

Простейшая схема несинхронизируемого T-триггера представлена на рис. 3.10, а. В этой схеме поступление сигнала  $T=1$  приводит к записи в двухступенчатый RS-триггер состояния, противоположного ранее хранимому. При этом, так как триггер двухступенчатый, на его выходе сигнал изменится только по завершении действия сигнала  $T=1$ , что исключает возникновение генерации в схеме с обратной связью. Можно считать, что в данной схеме единичный входной сигнал представляется спадом сигнала  $T=1$ , так как при любой продолжительности сигнала  $T=1$  изменение состояния T-триггера происходит только 1 раз — при снятии сигнала  $T=1$  (рис. 3.10, б).

При необходимости представлять потенциалом последовательность единиц на входе T-триггера используется синхронизируемая схема (рис. 3.10, в, г). Здесь единичный входной сигнал представляется высоким уровнем сигнала T при  $C=1$ . Поэтому высоким уровнем сигнала T можно представить последовательность 1 (рис. 3.10, д). Запись в триггер происходит при  $C=1$ , причем смена состояния происходит после окончания действия сигнала синхронизации  $C=1$ . При  $T=1$  со-

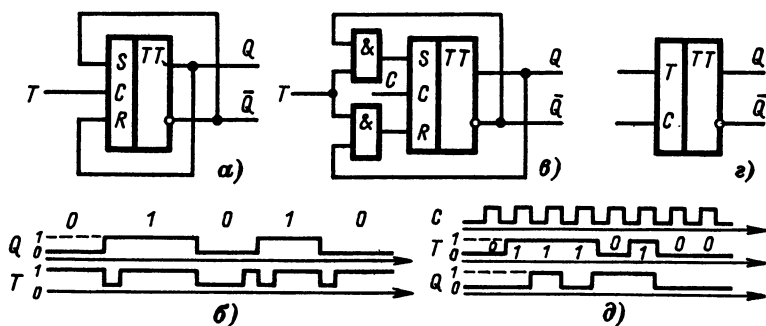


Рис. 3.10. T-триггер:

а — несинхронизируемый T-триггер; б — временная диаграмма работы несинхронизируемого T-триггера; в — синхронизируемый T-триггер; г — условное обозначение синхронизируемого T-триггера; д — временная диаграмма работы синхронизируемого T-триггера

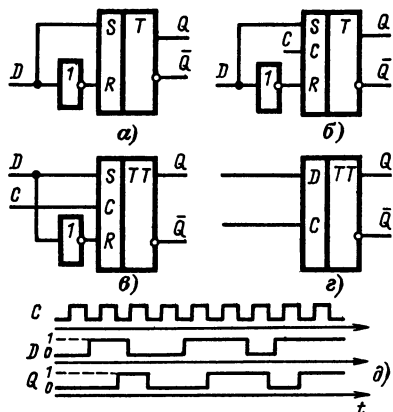


Рис. 3.11. *D*-триггер:

*a* — несинхронизируемый *D*-триггер; *б* — синхронизируемый одноктактный *D*-триггер; *в* — двухтактный *D*-триггер; *г* — условное обозначение двухтактного *D*-триггера; *д* — временная диаграмма работы двухтактного *D*-триггера

стояние триггера изменяется на противоположное, а при  $T=0$  оно не меняется. Временная диаграмма, поясняющая работу синхронизируемого *T*-триггера, показана на рис. 1.10, *д*.

*D*-триггер. Одним из самых широко употребляемых триггеров является *D*-триггер, который реализует функцию временной задержки. *D*-триггер имеет только режимы установки 1 и 0. В связи с этим несинхронизируемый *D*-триггер (рис. 3.11, *а*) не применяется, так как на его выходе будет просто повторяться входной сигнал. Синхронизируемый одноктактный *D*-триггер (рис. 3.11, *б*) задерживает распространение входного сигнала на время паузы между синхросигналами (задержка на полупериод). Для задержки на период (на один такт) используется двухтактный *D*-триггер.

Вариант построения двухтактного *D*-триггера показан на рис. 3.11, *в*. Под действием синхросигнала информация, поступающая на

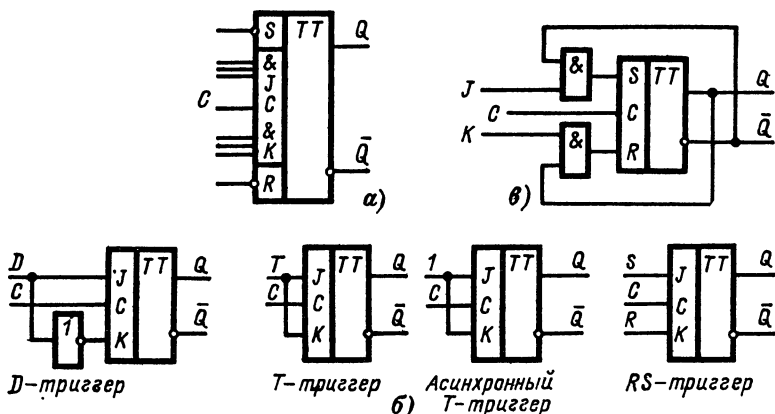


Рис. 3.12. *JK*-триггер:

*а* — условное обозначение двухтактного *JK*-триггера; *б* — способы использования *JK*-триггера; *в* — схема *JK*-триггера

Т а б л и ц а 3.3. Таблица переходов  $JK$ -триггера

$t$		$t+1$	Примечание
$J$	$K$	$Q$	
0	0	$Q(t)$	Хранение
0	1	0	Установка 0
1	0	1	Установка 1
1	1	$\bar{Q}(t)$	Инверсия

выход  $D$ , принимается в  $RS$ -триггер, но на выходе  $Q$  появляется с задержкой на такт:

$$Q(t+1) = D(t). \quad (3.7)$$

**$JK$ -триггер.** Распространенным типом триггера в системе интегральных логических элементов является двухтактный  $JK$ -триггер, условное обозначение которого показано на рис. 3.12, а.

У рассматриваемого триггера имеются входы несинхронизируемой установки  $R$  и  $S$ , с помощью которых при  $C=0$  триггер может быть установлен в состояние 1 путем подачи  $R=1$  и  $S=0$  либо в состояние 0 путем подачи  $R=0$  и  $S=1$ . При подаче сигналов  $R=S=1$ , не меняющих состояние схемы, работа триггера осуществляется под воздействием синхронизирующих входных сигналов. В этом случае функционирование триггера может быть описано табл. 3.3, причем такты  $t$  здесь, как и ранее, задаются сигналом  $C=1$ .

Здесь  $J=J_1J_2J_3$  и  $K=K_1K_2K_3$ . Входы  $J$  и  $K$  соответствуют входам установки в 1 и 0 триггера. Однако в отличие от  $RS$ -триггера в  $JK$ -триггере сигналы 1 могут одновременно поступить на входы  $J$  и  $K$ , при этом состояние триггера изменяется на противоположное, т. е. при  $J=K$  схема ведет себя как триггер со счетным входом.

Функцию переходов  $JK$ -триггера можно представить в виде булевой функции

$$Q(t+1) = \bar{K}(T) Q(t) \vee J(t) \bar{Q}(t) \quad (3.8)$$

при условии, что  $RS=1$ .

$JK$ -триггер удобен тем, что при различных вариантах подключения его входов можно получить схемы, функционирующие как  $RS$ -,  $D$ - и  $T$ -триггеры (рис. 3.12, б). Вариант построения схемы  $JK$ -триггера дан на рис. 3.12, в.

Во всех рассмотренных выше триггерах синхронизация (выделение) принимаемых воздействий производится единичным или нулевым значением сигнала синхронизации, представленным высоким или низким уровнем сигнала. Это означает, что принимаемое воздействие представляется сигналом, сохраняющим свое состояние на время существования уровня действующего значения сигнала синхронизации. Такая синхронизация называется статической, а триггеры — соответственно триггерами со статической синхронизацией. Помимо статической в настоящее время часто используется динамическая синхронизация.

При динамической синхронизации выделение принимаемых воздействий происходит не уровнем, а фронтом или срезом сигнала синхронизации. В этом случае принимаемое воздействие представляется сигналом, сохраняющим свое состояние на время существования фронта или среза сигнала синхронизации. Практически это означает, что принимаемое



воздействие должно быть установлено до момента изменения значения сигнала синхронизации (время предустановки оговаривается) и сохраняется неизменным все время фронта или среза (время удержания также оговаривается). Триггеры с динамической синхронизацией просты в реализации и могут использоваться вместо двухтактных, но в условиях строгой идентичности работы отдельных триггеров, обменивающихся между собой сигналами. Использование интегральной технологии обеспечивает высокий уровень идентичности схем, и в связи с этим триггеры с динамической синхронизацией широко применяются в схемотехнике СИС и БИС. Более подробно ознакомиться с принципами построения таких триггеров можно в [4].

### 3.5. Регистры

*Регистром* называется устройство, предназначенное для запоминания слова, а также для выполнения над словом некоторых логических преобразований. Регистр представляет собой совокупность триггеров, число которых соответствует числу разрядов в слове, и вспомогательных схем, обеспечивающих выполнение некоторых операций, среди которых могут быть:

- установка регистра в 0 («сброс»);
- прием слова из другого устройства (регистра, сумматора и т. д.);
- выдача слова из регистра (используется в схемах, предназначенных для работы на общий тракт передачи информации);
- сдвиг слова вправо или влево на требуемое число разрядов;
- преобразование последовательного кода слова в параллельный и наоборот;
- поразрядные логические операции.

На рис. 3.13 показана схема приема информации в регистр с использованием парафазной передачи информации, при которой на одном из входов триггера обязательно присутствует сигнал 1, устанавливающий триггер в нужное состояние независимо от той информации, которая в нем хранилась.

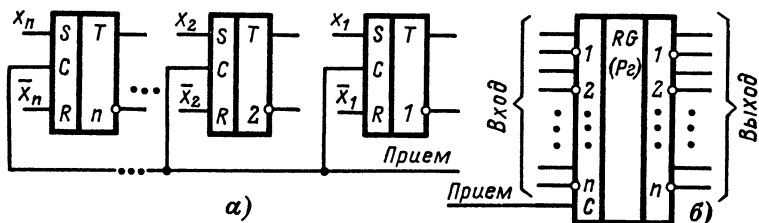


Рис. 3.13. Прием информации в регистр в парафазном коде:  
 а — функциональная схема регистра; б — условное обозначение регистра

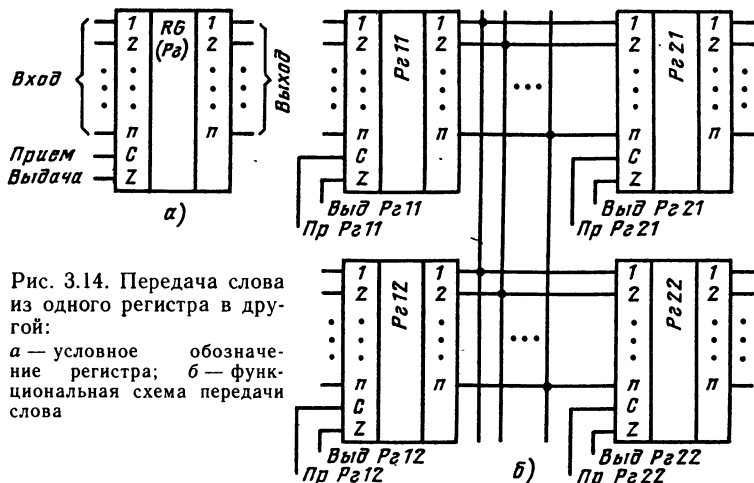


Рис. 3.14. Передача слова из одного регистра в другой:

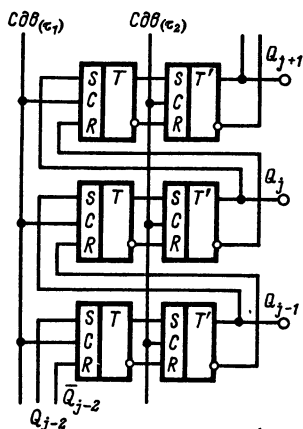
а — условное обозначение регистра; б — функциональная схема передачи слова

Регистр на  $D$ -триггерах позволяет принимать информацию без парафазного представления сигналов.

Регистры с отключаемыми выходами имеют дополнительный вход, устанавливающий режим выдачи кода с регистра или режим отключения регистра. Условное обозначение такого регистра показано на рис. 3.14, а.

Как правило, операция передачи слова с регистра выполняется в схемах с несколькими источниками информации, порознь подключенными к одному общему тракту передачи информации, и совмещается с операцией приема этого слова в регистр-приемник. Это показано на рис. 3.14, б, где слово передается из регистра  $R_{г11}$  в регистр  $R_{г22}$  под воздействием сигналов  $ВыдR_{г11}$  и  $ПрR_{г22}$ . Отсутствие сигнала  $ВыдR_{г12}$  обеспечивает отключение от тракта передачи  $R_{г12}$ . При подаче сигналов  $ВыдR_{г12}$  и  $ПрR_{г22}$  осуществляется передача сигнала из  $R_{г12}$  в  $R_{г22}$  и т. п.

Рис. 3.15. Сдвигающий регистр на одноктактных  $RS$ -триггерах



Операция сдвига кода — это перемещение в регистре всех разрядов кода слова на одинаковое число разрядов влево или вправо. В этом случае разряды слова, вышедшие из разрядной сетки регистра влево (или вправо), теряются, а в освободившиеся при сдвиге разряда регистра записываются нули.

Функциональная схема сдвигающего регистра на  $RS$ -триггерах приведена на рис. 3.15.

Сигналом  $Cдв(\tau_1)$  код в регистре передается со сдвигом с выходных (основных) триггеров на входные (дополнительные) триггеры соседних разрядов, а потом сигналом  $Cдв(\tau_2)$  переписывается с входных триггеров на выходные триггеры в тех же самых разрядах. Сигналы  $Cдв(\tau_1)$  и  $Cдв(\tau_2)$  во времени не совпадают.

### 3.6. Счетчики

*Счетчиком* называется узел ЭВМ, предназначенный для подсчета числа входных сигналов. Счетчики используются в ЭВМ для образования последовательностей адресов команд, для подсчета числа циклов выполнения операций и т. п.

Счетчики принято подразделять на суммирующие, вычитающие и реверсивные.

На рис. 3.16 приведены схема несинхронизируемого четырехразрядного двоичного суммирующего счетчика с последовательным переносом и временная диаграмма его работы. Таблица 3.4 показывает состояния, в которых находятся триггеры счетчика при воздействии серии входных сигналов  $x_{сч}$ .

Здесь на входы  $J$  и  $K$   $JK$ -триггеров подаются сигналы 1. Выход каждого предыдущего триггера  $Q_{n-1}$  соединен со входом синхронизации  $C_n$  последующего триггера. Каждый  $JK$ -триггер в счетчике выполняет функцию несинхронизируемого триггера со

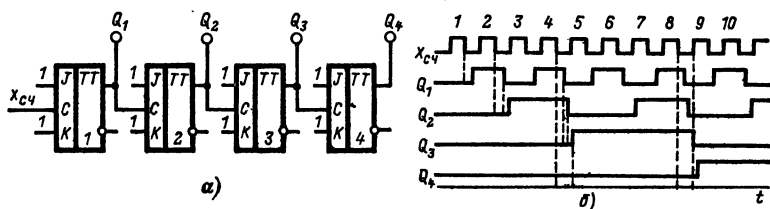


Рис. 3.16. Несинхронизируемый двоичный счетчик с последовательным переносом:

а — функциональная схема; б — временная диаграмма

Т а б л и ц а 3.4. Таблица состояний двоичного счетчика

$x_{сч}$	$Q_4$	$Q_3$	$Q_2$	$Q_1$	$x_{сч}$	$Q_4$	$Q_3$	$Q_2$	$Q_1$
0	0	0	0	0	8	1	0	0	0
1	0	0	0	1	9	1	0	0	1
2	0	0	1	0	10	1	0	1	0
3	0	0	1	1	11	1	0	1	1
4	0	1	0	0	12	1	1	0	0
5	0	1	0	1	13	1	1	0	1
6	0	1	1	0	14	1	1	1	0
7	0	1	1	1	15	1	1	1	1

счетным входом. По спаду единичного входного сигнала изменяется состояние триггера младшего разряда счетчика на противоположное (т. е. реализуется сложение по модулю 2 в этом разряде). В последующих разрядах аналогичное действие производит сигнал переноса.

Обычно счетчик имеет цепь установки в нулевое состояние (сброс триггеров в 0). Однако начальное состояние счетчика необязательно нулевое. Начальное состояние может устанавливаться передачей в счетчик кода некоторого числа, и с него уже будет начинаться операция счета единиц. Такой режим работы счетчика необходим, например, при образовании последовательности адресов команд при заданном исходном адресе. С ростом разрядности счетчика понижается предельная частота его работы. Это объясняется тем, что с ростом разрядности счетчика  $n$  будет возрастать задержка поступления сигнала на вход  $C$  некоторого  $j$ -го разряда относительно времени поступления входного сигнала  $x_{сч}$  на вход  $C$  младшего разряда счетчика. Из временной диаграммы видно, что такая задержка может привести к искажению информации в счетчике (моменты времени 4 и 8). Для повышения быстродействия счетчики выполняют с параллельным переносом.

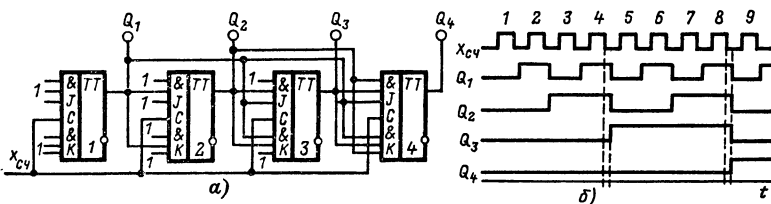


Рис. 3.17. Синхронизируемый двоичный счетчик с параллельным переносом:

а — функциональная схема; б — временная диаграмма



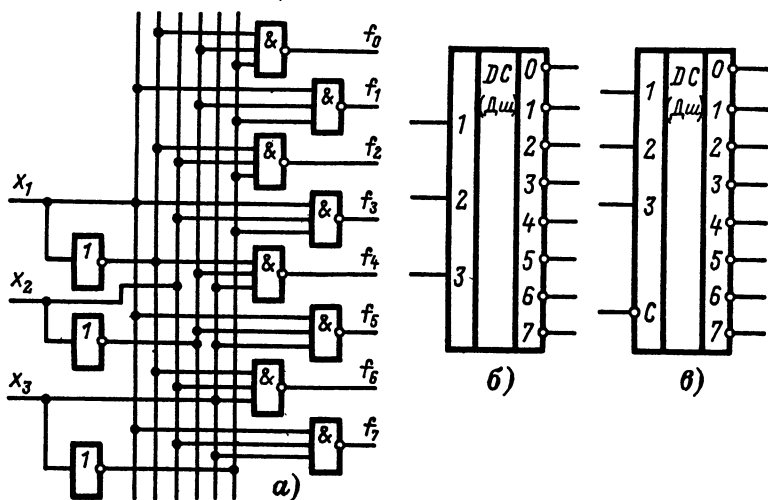


Рис. 3.18. Дешифратор с инверсными выходами:

*а* — функциональная схема; *б* — условное обозначение дешифратора; *в* — условное обозначение дешифратора со входом синхронизации

слова, находящегося в регистре (в счетчике), в управляющий сигнал на одном из выходов дешифратора.

На рис. 3.18, *а* показан способ построения дешифратора на примере схемы дешифратора с инверсными выходами для трехразрядного входного слова. Схема представляет собой набор из восьми трехходовых элементов И — НЕ, на входы которых поданы все возможные комбинации прямых и инверсных значений разрядов слова.

Если каждый выходной элемент дешифратора имеет вход для сигнала синхронизации, то такая схема называется дешифратором со входом синхронизации (рис. 3.18, *в*). При  $C=1$  все выходы данного дешифратора будут иметь единичное значение, а при  $C=0$  он будет работать так же, как и схема на рис. 3.18, *б*. Такого вида схемы выпускаются в составе комплексов интегральных логических элементов.

Из логических элементов, являющихся дешифраторами, можно строить дешифраторы на большее число входов, при этом, как правило, используются дешифраторы с инверсными выходами и входами синхронизации. Каскадное включение таких схем позволяет легко наращивать число дешифрируемых переменных. Принцип построения схем нетрудно понять, обратившись к рис. 3.19. Здесь показан дешифратор на четыре входа

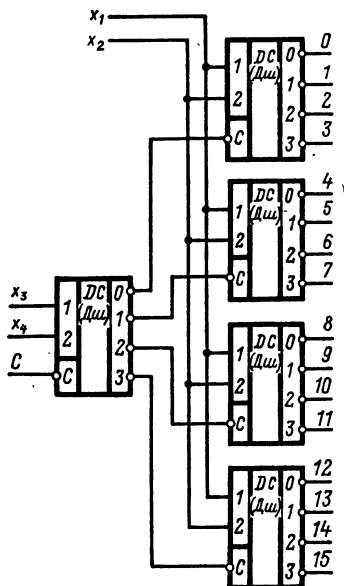


Рис. 3.19. Каскадный дешифратор

с инверсными выходами, построенный из интегральных элементов, реализующих схемы двухвходовых дешифраторов.

### 3.8. Мультиплексоры

Мультиплексором называется схема, осуществляющая передачу сигналов с одной из входных линий в выходную. Выбор входной (информационной) линии производится кодом, поступающим на управляющие входы мультиплексора. Мультиплексор с  $k$  управляющими входами  $x_{k-1}, x_{k-2}, \dots, x_1, x_0$  имеет  $2^k$  информационных входов  $D_0, D_1, \dots, D_{2^k-1}$ .

Набор сигналов, поступающих на управляющие входы, задает двоичное число вида

$$X^* = x_{k-1}^* 2^{k-1} + x_{k-2}^* 2^{k-2} + \dots + x_1^* 2^1 + x_0^*,$$

где  $x_i^* = (0, 1)$  — значение сигнала на входе  $x_i$ . Выходной сигнал мультиплексора повторяет сигнал информационного входа  $D$  с номером  $X^*$ .

Работа мультиплексора может иллюстрироваться контактной схемой (рис. 3.20, а). Условное обозначение мультиплексора, коммутирующего  $2^k$  линий, показано на рис. 3.20, б. Пример функциональной схемы мультиплексора с двумя управляющими входами дан на рис. 3.20, в.

Функция, реализуемая мультиплексором, может быть представлена в виде

$$y = \bar{x}_{k-1} \bar{x}_{k-2} \dots \bar{x}_0 D_0 \vee \bar{x}_{k-1} \bar{x}_{k-2} \dots$$

$$\dots x_0 D_1 \vee \dots \vee x_{k-1} x_{k-2} \dots x_0 D_{2^k-1} =$$

$$= R_0 D_0 \vee R_1 D_1 \vee \dots \vee R_j D_j \vee \dots \vee R_{2^k-1} D_{2^k-1}, \quad (3.9)$$

где  $R_j$  — конъюнкция, равная 1 на наборе значений переменных

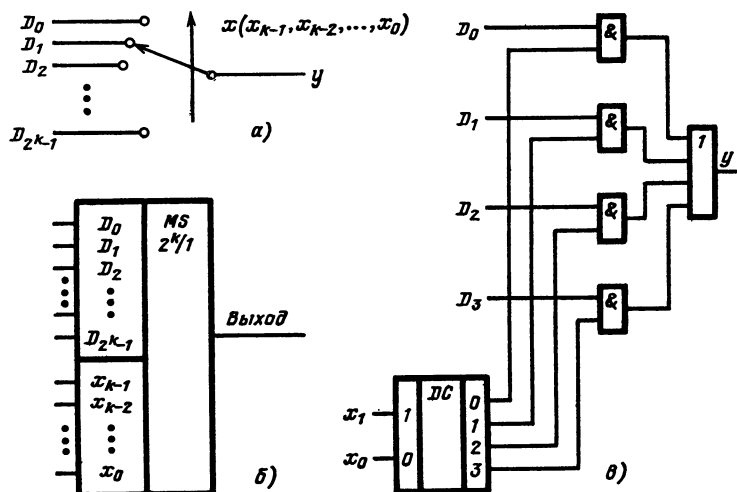


Рис. 3.20. Мультиплексор:

*a* — контактная модель; *б* — условное обозначение; *в* — функциональная схема

$x_{k-1} x_{k-2} \dots x_0$  с номером  $j$  (т. е. на наборе, представляющем в двоичном виде число  $j$ ).

Функция эта нелинейна, немонотонна и несамоудовлетворительна и согласно теореме об ослабленной функциональной полноте позволяет при использовании булевых констант 1 и 0 путем суперпозиций реализовать любую булеву функцию [15].

Синтез комбинационных схем на мультиплексорах удобно осуществлять с помощью теоремы разложения [15]. Эта теорема позволяет любую булеву функцию  $F(x_{n-1}, x_{n-2}, \dots, x_1, x_0)$  разложить по  $i$  переменным  $x_{n-1}, x_{n-2}, \dots, x_{n-i}$  и представить как

$$\begin{aligned}
 F &= \bar{x}_{n-1} \bar{x}_{n-2} \dots \bar{x}_{n-i} F^0 \vee \\
 &\vee \bar{x}_{n-1} \bar{x}_{n-2} \dots x_{n-i} F^1 \vee \dots \\
 &\dots \vee x_{n-1} x_{n-2} \dots x_{n-i} F^{2^i-1} = \\
 &= R_0 F^0 \vee R_1 F^1 \vee \dots \vee R_j F^j \vee \dots \vee R_{2^i-1} F^{2^i-1}, \quad (3.10)
 \end{aligned}$$

где  $R_j$  — конъюнкция, равная 1 на наборе значений переменных  $x_{n-1}, x_{n-2}, \dots, x_{n-i}$  с номером  $j$ , а  $F^j$  — функция, полученная из  $F$  подстановкой в нее набора значений переменных  $x_{n-1}, x_{n-2}, \dots, x_{n-i}$  с номером  $j$ .



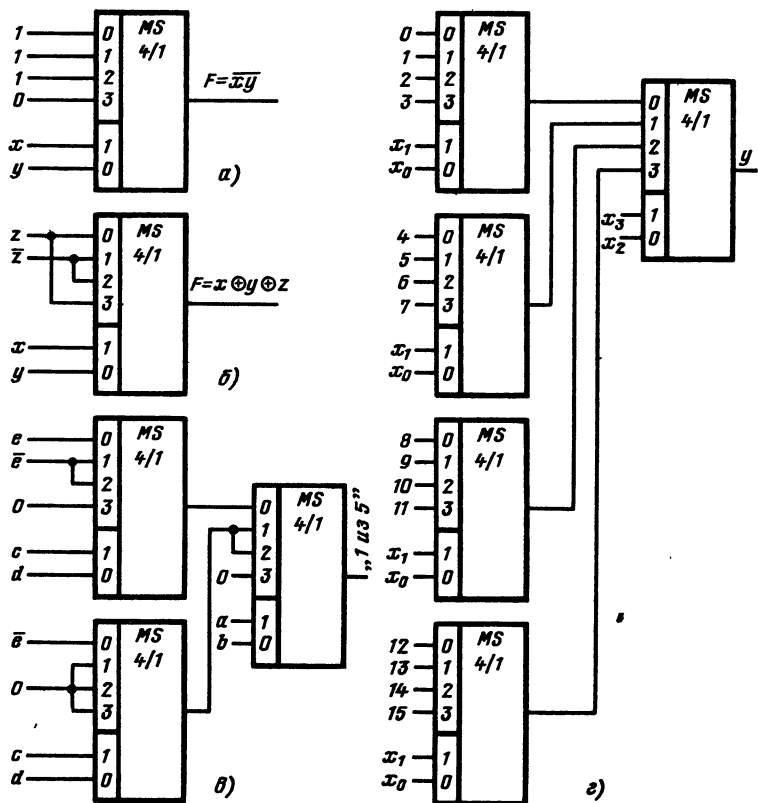


Рис. 3.21. Схемы из мультиплексоров:

а — И — НЕ; б — сложение по модулю 2; в — двухкаскадное мультиплексирование; г — многокаскадный мультиплексор

Сравнивая (3.9) и (3.10), легко заметить, что на основе разложения булевой функции  $F$  по  $k$  переменным можно строить реализующую ее схему на мультиплексорах. Для этого необходимо подать на входы мультиплексора  $D_0, D_1, \dots, D_{2^k-1}$  сигналы, представляющие функции разложения  $F^0, F^1, \dots$ , а на управляющие входы — сигналы  $k$  переменных, по которым выполнялось разложение. Если функция  $F$  имеет  $n = k$  аргументов, то функции разложения  $F^0, F^1, \dots$  — константы, и на вход  $D_j$  мультиплексора подается константа, равная значению  $F$  на наборе с номером  $j$ . При числе аргументов функции  $F$ , равном  $(k+1)$ , функции разложения могут иметь вид  $x$ , либо  $\overline{x}$ , либо 0, либо

1. Подавая на входы  $D_0, D_1 \dots$  мультиплексора такие сигналы, можно реализовать на одной микросхеме мультиплексора любую функцию  $k+1$  переменной. Если  $n > k+1$ , то функции разложения могут зависеть от нескольких переменных и для реализации их потребуются схемы, которые можно построить, разложив функции разложения так же, как и исходную по  $k$  переменным, и т. д.

На рис. 3.21 даны примеры построенных на мультиплексорах схем, реализующих функции И — НЕ, сложения по модулю 2 и функцию выделения наборов пяти переменных, содержащих одну 1.

Корпус (кристалл) микросхемы может содержать несколько мультиплексорных схем. Выход мультиплексора может быть как прямым, так и инверсным по отношению к сигналам  $D_0, D_1 \dots$ ; мультиплексор может быть снабжен входом выборки (синхронизации). Число управляющих входов у микросхем мультиплексоров обычно находится в пределе 1—4. При необходимости построения мультиплексоров с большим числом входов строятся каскадные схемы. Пример такой схемы дан на рис. 3.21, г.

### 3.9. Сумматоры

*Сумматором* называется узел ЭВМ, выполняющий арифметическое суммирование кодов чисел. Обычно сумматор представляет собой комбинацию одноразрядных суммирующих схем.

При сложении двух чисел в каждом разряде производится сложение трех цифр: цифры данного разряда первого слагаемого, цифры данного разряда второго слагаемого и цифры (1 или 0) переноса из соседнего младшего разряда. В результате сложения для каждого разряда получаются цифра суммы для этого разряда и цифра (1 или 0) переноса в следующий, старший разряд.

В табл. 3.5 приведены варианты, возникающие при сложении двух двоичных чисел.

По табл. 3.5 можно составить булевы функции для описания одноразрядного сумматора — устройства, вырабатывающего на выходе сигналы суммы и переноса при поступлении на входы двух цифр слагаемых и цифры переноса из предыдущего, младшего разряда:

$$S_i = a_i \bar{b}_i \bar{P}_i \vee \bar{a}_i b_i \bar{P}_i \vee \bar{a}_i \bar{b}_i P_i \vee a_i b_i P_i; \quad (3.11)$$

$$P_{i+1} = a_i b_i \bar{P}_i \vee a_i \bar{b}_i P_i \vee \bar{a}_i b_i P_i \vee a_i b_i P_i, \quad (3.12)$$

где  $a_i, b_i$  — цифры слагаемых в данном разряде;  $P_i$  — цифра

Таблица 3.5

Цифры переноса из предыдущего разряда $P_i$	Первое слагаемое $a_i$	Второе слагаемое $b_i$	Сумма $S_i$	Цифра переноса в старший разряд $P_{i+1}$
0	0	0	0	0
0	1	0	1	0
0	0	1	1	0
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1
1	0	1	0	1
1	1	1	1	1

переноса из предыдущего (младшего) разряда;  $S_i$  — сумма;  $P_{i+1}$  — цифра переноса в старший разряд.

Выражение для цифры переноса в следующий разряд может быть приведено к более простому виду:

$$P_{i+1} = a_i b_i \vee a_i P_i \vee b_i P_i. \quad (3.13)$$

Преобразуя с помощью правил булевой алгебры выражения для цифры суммы и цифры переноса, можно получать различные соотношения, которым будут соответствовать варианты построения схем полных сумматоров.

Функциональная схема одноразрядного комбинационного сумматора, реализующего соотношения (3.11) и (3.13), показана на рис. 3.22.

Параллельный (многоразрядный) сумматор может быть составлен из одноразрядных сумматоров, число которых равно числу разрядов слагаемых, путем соединения выхода, на котором формируется сигнал переноса данного разряда, со входом для сигнала переноса соседнего старшего разряда (см. рис. 3.23).

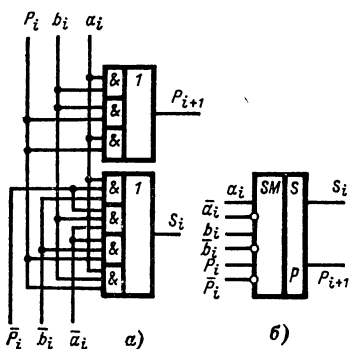


Рис. 3.22. Комбинационный одноразрядный сумматор:  
а — функциональная схема; б — условное обозначение

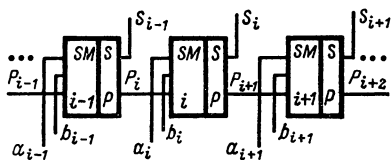


Рис. 3.23. Параллельный сумматор с последовательным переносом

После того как выработан результат сложения на выходах комбинационных схем формирования суммы, он обычно запоминается в отдельном триггерном регистре.

Быстродействие одnorазрядного комбинационного сумматора характеризуется временем установления выходных сигналов суммы и переноса после установки сигналов на входах сумматора. Наиболее важным является время распространения сигнала переноса в одnorазрядном сумматоре, так как при образовании многоразрядного сумматора из одnorазрядных схем сигнал переноса может распространяться от разряда к разряду. Это время определяется временами задержек в логических элементах и количеством последовательно включенных элементов в схеме распространения сигнала переноса.

По характеру распространения переноса различают следующие виды сумматоров: с поразрядным последовательным переносом, с параллельным (одновременным) переносом, с групповым переносом.

*Сумматоры с поразрядным последовательным переносом.* В сумматорах этого типа (рис. 3.23) перенос распространяется последовательно от разряда к разряду по мере образования цифры суммы в каждом отдельном разряде. При наиболее неблагоприятных условиях для распространения переноса, например при сложении чисел  $11\dots 11$  и  $00\dots 001$ , произойдет «пробег» 1 переноса через сумматор от самого младшего разряда к самому старшему. Поэтому в худшем случае время распространения переноса

$$T_{\text{пер}} = \tau_1 n, \quad (3.14)$$

где  $\tau_1$  — время распространения переноса в одном разряде;  $n$  — число разрядов сумматора. Данный тип сумматора наиболее прост с точки зрения схемы цепей распространения переноса, но имеет сравнительно низкое быстродействие.

*Сумматоры с параллельным переносом.* Можно построить сумматор, в котором сложение выполняется как поразрядная операция и на распространение переноса не требуется дополнительного времени.

Затраты оборудования на построение сумматора такого типа, особенно при большом числе разрядов, настолько велики, что в чистом виде он практически не находит применения. Принцип параллельного формирования переноса используется в сумматорах с групповым переносом.

*Сумматоры с групповым переносом.* Сумматор разбивается на несколько групп примерно равной длины. Сигнал переноса, поступающий на вход младшего разряда группы, при наличии

условий распространения переноса во всех разрядах данной группы передается на вход младшего разряда соседней, более старшей группы в обход данной группы.

Схема формирования сигнала переноса в младшем разряде каждой группы дополняется для этой цели схемой И, реализующей булеву функцию

$$P_{\text{уск}} = P_i c_i c_{i+1} \dots c_{i+k-1}, \quad (3.15)$$

где  $P_{\text{уск}}$  — сигнал ускоренного переноса;  $P_i$  — сигнал переноса в младший разряд группы, содержащий  $k$  разрядов;  $c_i, \dots, c_{i+k-1}$  — условия распространения переноса в разрядах групп ( $c_i = \bar{a}_i b_i \vee a_i \bar{b}_i$ ).

В таком сумматоре максимальная задержка распространения переноса определяется задержкой его в младшей и старшей группах, а также в цепях обхода остальных групп.

Максимальная задержка сигнала переноса может быть уменьшена, если при разбиении сумматора на группы использовать параллельное (одновременное) формирование переноса внутри групп.

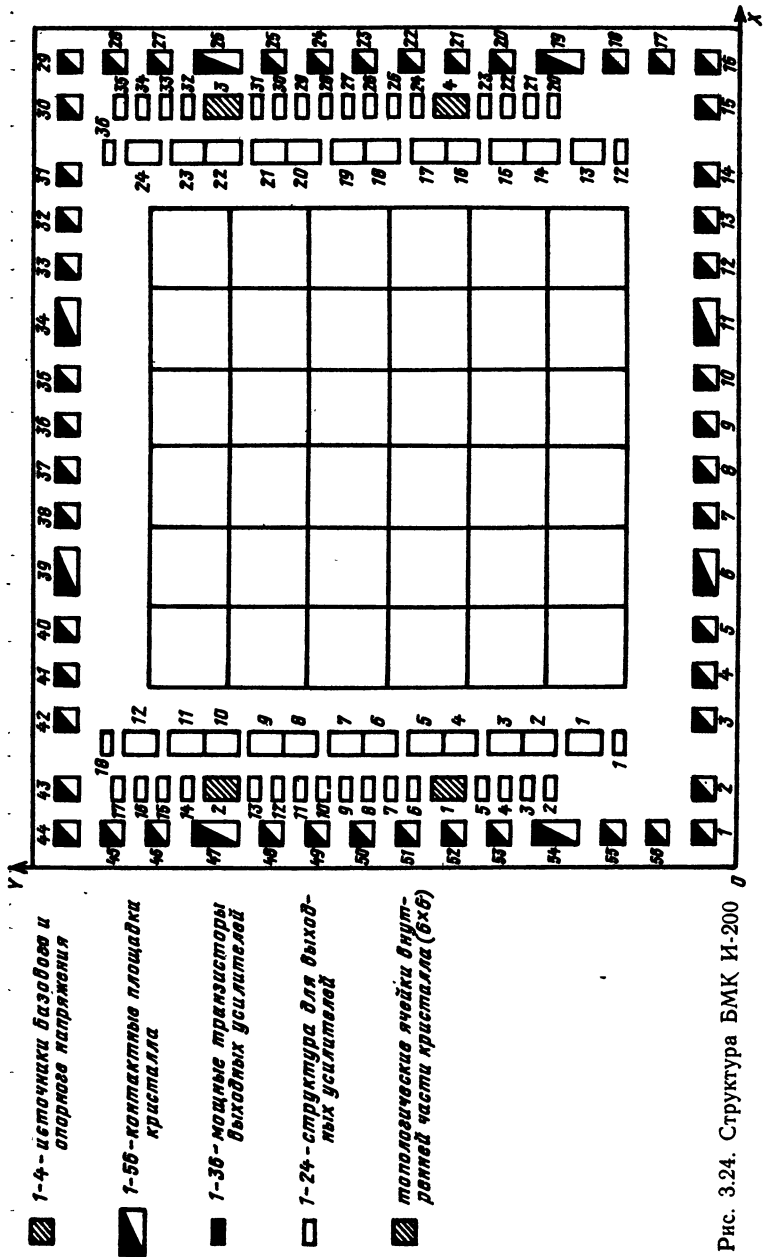
### 3.10. Матричные БИС

Улучшение технико-экономических характеристик ЭВМ (повышение быстродействия, надежности, снижение габаритных размеров, стоимости и т. д.) достигается за счет все более широкого применения БИС, при этом возникает следующая ситуация: рост степени интеграции схем сопровождается ростом их специализации. Создание достаточно универсальных по применению микропроцессоров частично решает проблему применимости БИС, но оставляет проблему построения БИС для схем, которые не могут быть реализованы на базе микропроцессоров, например, из-за необходимости обеспечения высокого быстродействия их работы.

Создание специализированных БИС и СБИС при относительно небольших объемах производства может быть экономически выгодным только в том случае, если сроки и стоимость их проектирования будут малы.

Добиться этого удастся при проектировании быстродействующих БИС на основе *базовых матричных кристаллов* (БМК). Такие БИС носят название *матричных* (МаБИС) и широко применяются в схемотехнике современных ЭВМ.

Базовый матричный кристалл представляет собой полупроводниковый кристалл, содержащий матрицу базовых ячеек, расположенную в центральной части, и группу буферных ячеек,



расположенных по периферии кристалла. В ячейках БМК в определенном установленном заранее порядке размещаются несвязные компоненты: транзисторы, диоды, резисторы и т. п. Состав ячейки БМК позволяет построить разнообразные логические схемы, называемые функциональными элементами ячеек БМК. Набор стандартных решений по построению схем функциональных элементов ячеек БМК образует *библиотеку ячеек БМК*. Как правило, библиотека является расширяемой. Базовые ячейки служат для реализации внутренних схем БИС, а периферийные — для построения схем входов и выходов. Ячейки отделяются на кристалле друг от друга пространством, предназначенным для размещения металлизированных соединений, выполняемых в МаБИС с помощью одного или нескольких слоев металлизации. Пример структуры БМК дан на рис. 3.24 [50].

Базовые матричные кристаллы служат основой для создания различных МаБИС, многообразие которых определяется различными вариантами межсоединений, формируемых на последнем этапе технологического процесса изготовления МаБИС. Эффективность использования БМК при создании БИС для ЭВМ объясняется следующими причинами. Один и тот же БМК используется для разработки большого числа БИС; для разработки МаБИС широко используются стандартные (библиотечные) решения по построению отдельных подсхем; разрабатываются только схемы межсоединений ячеек БМК и соответствующие фотошаблоны; разработка ведется с помощью систем автоматизации проектирования (САПР).

Матричные БИС, применяемые в настоящее время на ЭВМ, реализуются на БМК различных технологий. Наиболее быстродействующими являются матричные ЭСЛ БИС с элементами, имеющими задержку, измеряемую в долях наносекунд. Количество элементов в БМК может составлять несколько десятков тысяч. Количество выводов МаБИС зависит от типа используемого для БМК корпуса и может составлять сотни выводов. В качестве примера укажем, что МаБИС серии K1520XM2 строятся на основе БМК, заключенного в корпусе со 108 выводами, с элементами типа ЭСЛ, имеющими задержку 1 нс. Уровень интеграции БМК оценивается в 1100 эквивалентных логических элементов 2И — НЕ (вентилей).

Матричные БИС получили широкое распространение в качестве элементной базы ЭВМ четвертого поколения.

### Контрольные вопросы

1. В чем основное отличие импульсного способа представления информации?

2. Приведите пример преобразования информации, не выполняемого комбинационными схемами.

3. В каких случаях допускается объединение выходов логических элементов?

4. Сравните по быстродействию различные типы интегральных элементов.

5. Полагая, что в Вашем распоряжении имеются двухступенчатый *D*-триггер (см. рис. 3.11, *г*) и элементы И — НЕ (см. рис. 3.5, *д*), постройте схему *JK*-триггера.

6. Постройте трехрегистровую схему только с одним общим трактом передачи информации, в которой можно выполнять передачу кодов в любой паре регистров. В каждый данный момент времени источником может быть любой, но только один регистр.

7. Сколько выходов может иметь двухкаскадный дешифратор, построенный из схем, приведенных на рис. 3.16, *в*?

8. Из мультиплексоров с двумя управляющими входами постройте схему выделения наборов пяти переменных с двумя единицами.

9. Постройте схему синхронизируемого двоичного счетчика из двухступенчатых *D*-триггеров и элементов И — НЕ.

10. Чем объясняется распространение БИС и СБИС, строящихся на основе БМК?

## Глава 4

# ПРИНЦИПЫ ПОСТРОЕНИЯ УСТРОЙСТВ ПАМЯТИ

## 4.1. Общие сведения и классификация устройств памяти

*Памятью ЭВМ* называется совокупность устройств, служащих для запоминания, хранения и выдачи информации. Отдельные устройства, входящие в эту совокупность, называют *запоминающими устройствами* или *памятями* того или иного типа.

Оба эти термина в настоящее время стали почти синонимами. Однако термин «запоминающее устройство» (ЗУ) обычно употребляют, когда речь идет о принципе построения некоторого устройства памяти (например, полупроводниковые ЗУ, ЗУ на магнитных дисках и т. д.), а термин «память» — когда хотят подчеркнуть выполняемую устройством памяти логическую функцию или место расположения в составе оборудования ЭВМ (например, оперативная память, внешняя память и т. д.).

Производительность и вычислительные возможности ЭВМ в значительной степени определяются составом и характеристиками ее ЗУ. В составе ЭВМ используется одновременно несколько типов ЗУ (несколько типов памяти), отличающихся принципом действия, характеристиками и назначением.



Основными операциями в памяти в общем случае являются занесение информации в память — *запись* и выборка информации из памяти — *считывание*. Обе эти операции называются *обращением к памяти*, или, подробнее, *обращением при считывании и обращением при записи*.

При обращении к памяти производится считывание или запись некоторой единицы данных — различной для устройств разного типа. Такой единицей может быть, например, байт, машинное слово или блок данных.

Важнейшими характеристиками отдельных устройств памяти (запоминающих устройств) являются емкость памяти, удельная емкость, быстродействие.

*Емкость памяти* определяется максимальным количеством данных, которые могут в ней храниться. Емкость измеряется в двоичных единицах (битах), машинных словах, но большей частью в байтах (1 байт = 8 бит), при этом часто емкость памяти выражают через число  $K=1024$ : Кбит (килобит), Кслов (килослов) или Кбайт (килобайт), при этом 1024 Кбайт обозначают как 1 Мбайт (мегабайт).

*Удельная емкость* есть отношение емкости ЗУ к его физическому объему.

*Быстродействие памяти* определяется продолжительностью операции обращения, т. е. временем, затрачиваемым на поиск нужной единицы информации в памяти и на ее считывание (*время обращения при считывании*), или временем на поиск места в памяти, предназначенного для хранения данной единицы информации, и на ее запись в память (*время обращения при записи*).

Продолжительность обращения к памяти (время цикла памяти) при считывании

$$t_{\text{обр}}^{\text{счит}} = t_{\text{дост}}^{\text{счит}} + t_{\text{счит}}, \quad (4.1)$$

где  $t_{\text{дост}}^{\text{счит}}$  — время доступа, определяющееся промежутком времени между моментом начала операции обращения при считывании до момента, когда становится возможным доступ к данной единице информации;  $t_{\text{счит}}$  — продолжительность самого физического процесса считывания, т. е. процесса обнаружения и фиксации состояний соответствующих запоминающих элементов или участков поверхности носителя информации.

В некоторых устройствах памяти считывание информации сопровождается ее разрушением (стиранием). В таком случае цикл обращения должен содержать операцию восстановления (регенерации) считанной информации на прежнем месте в памяти.

Продолжительность обращения (время цикла) при записи

$$t_{\text{обр}}^{\text{зап}} = t_{\text{дост}}^{\text{зап}} + t_{\text{зап}}, \quad (4.2)$$

где  $t_{\text{дост}}^{\text{зап}}$  — время доступа при записи, т. е. время от момента начала обращения при записи до момента, когда становится возможным доступ к запоминающим элементам (или участкам поверхности носителя), в которые производится запись;  $t_{\text{подг}}$  — время подготовки, расходуемое на приведение в исходное состояние запоминающих элементов или участков поверхности носителя информации для записи определенной единицы информации (например, байта или слова);  $t_{\text{зап}}$  — время занесения информации, т. е. изменения состояния запоминающих элементов (участков поверхности носителя).

Большей частью

$$t_{\text{дост}}^{\text{счит}} = t_{\text{дост}}^{\text{зап}} = t_{\text{дост}}.$$

В качестве продолжительности цикла обращения к памяти принимается величина

$$t_{\text{обр}} = \max(t_{\text{обр}}^{\text{счит}}, t_{\text{обр}}^{\text{зап}}). \quad (4.3)$$

В зависимости от реализуемых в памяти операций обращения различают: а) память с произвольным обращением (возможны считывание и запись данных в память); б) память только для считывания информации («постоянная» или «односторонняя»). Запись информации в постоянную память производится в процессе ее изготовления или настройки<sup>1</sup>.

По способу организации доступа различают устройства памяти с непосредственным (произвольным), с прямым (циклическим) и последовательным доступами.

В памяти с *непосредственным (произвольным)* доступом время доступа, а поэтому и цикл обращения не зависят от места расположения участка памяти, с которого производится считывание или в который записывается информация. В большинстве случаев непосредственный доступ реализуется при помощи электронных (полупроводниковых) ЗУ. В подобных памяти цикл обращения обычно составляет 1 мкс или всего несколько сотен или десятков наносекунд. Число разрядов, считываемых или записываемых в памяти с непосредственным доступом параллельно во времени за одну операцию обращения, называется *шириной выборки*.

---

<sup>1</sup> Эти типы памяти соответствуют терминам RAM (random — access method — память с произвольным обращением) и ROM (read — only method — память только для считывания).

В двух других типах памяти используются более медленные электромеханические процессы. В устройствах *памяти с прямым доступом*, к которым относятся устройства с магнитными барабанами и дисками, благодаря непрерывному вращению носителя информации возможность обращения к некоторому участку носителя для считывания или записи циклически повторяется. В такой памяти время доступа составляет обычно от нескольких долей секунды до нескольких десятков миллисекунд.

В *памяти с последовательным доступом* производится последовательный просмотр участков носителя информации, пока нужный участок носителя не займет некоторое исходное положение. Характерным примером является ЗУ на магнитных лентах. Время доступа может в неблагоприятных случаях расположения информации достигнуть нескольких минут.

Запоминающие устройства различаются также по выполняемым в ЭВМ функциям, зависящим в частности, от места расположения ЗУ в структуре ЭВМ.

Требования к емкости и быстродействию памяти являются противоречивыми. Чем больше быстродействие, тем технически труднее достигается и дороже обходится увеличение емкости памяти. Стоимость памяти составляет значительную часть общей стоимости ЭВМ. Поэтому память ЭВМ организуется в виде иерархической структуры запоминающих устройств, обладающих различными быстродействием и емкостью (рис. 4.1). В общем случае ЭВМ содержит *сверхоперативную память* (СОП)

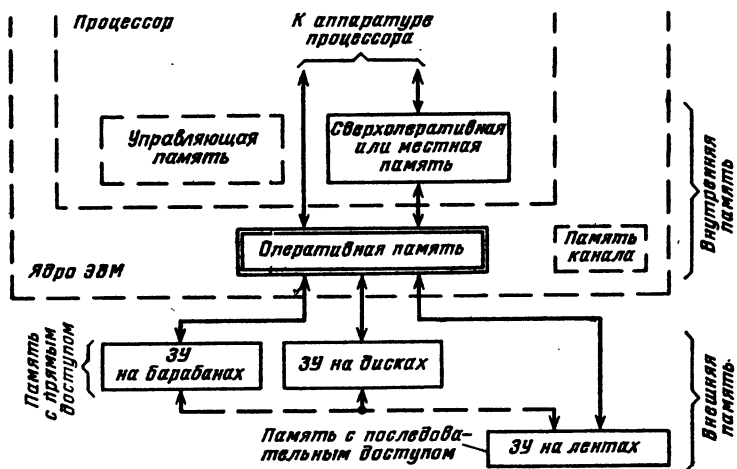


Рис. 4.1. Иерархическая структура памяти ЭВМ

или *местную память, оперативную или основную память (ОП), память с прямым доступом* на магнитных барабанах и на магнитных дисках, *память с последовательным доступом* на магнитных лентах. Порядок перечисления устройств соответствует убыванию их быстродействия и возрастанию емкости. Каждый уровень иерархии может содержать несколько экземпляров (модулей) соответствующих устройств для получения нужной емкости данного уровня памяти. На рис. 4.1 сплошными и штриховыми линиями показаны соответственно обычно и сравнительно редко реализуемые пути передачи данных между отдельными ступенями иерархической памяти. Иерархическая структура памяти позволяет экономически эффективно сочетать хранение больших объемов информации с быстрым доступом к информации в процессе обработки.

*Оперативной или основной памятью (ОП)* называют устройство, которое служит для хранения информации (данных программ, промежуточных и конечных результатов обработки), непосредственно используемой в процессе выполнения операций в арифметическо-логическом устройстве и устройстве управления процессора.

В процессе обработки информации осуществляется тесное взаимодействие процессора и ОП. Из ОП в процессор поступают команды программы и операнды, над которыми производятся предусмотренные командой операции, а из процессора в ОП направляются для хранения промежуточные и конечные результаты обработки.

Характеристики ОП непосредственно влияют на основные показатели ЭВМ и в первую очередь на скорость ее работы. Оперативная память высокопроизводительных ЭВМ имеет емкость несколько миллионов байт и цикл обращения около 0,5 мкс (и менее). Запоминающие устройства ОП, ранее выполнявшиеся на магнитных (ферритовых) сердечниках и тонких магнитных пленках, в настоящее время изготавливаются на интегральных микросхемах с большой степенью интеграции (полупроводниковые ЗУ).

В ряде случаев быстродействие ОП оказывается недостаточным, и в состав машины приходится включать СОП (буферную или кэш-память на несколько сотен или тысяч машинных слов с циклом обращения, составляющим несколько десятков наносекунд. Такие СОП выполняются на быстродействующих интегральных микросхемах. Быстродействие СОП должно соответствовать скорости работы арифметическо-логических и управляющих устройств процессора. Сверхоперативная (буферная) память используется для промежуточного хранения считываемых процессором из ОП участков программы и групп данных,

в качестве рабочих ячеек программы, индексных регистров, для хранения служебной информации, используемой при управлении вычислительным процессом. Она выполняет роль согласующего звена между быстродействующими логическими устройствами процессора и более медленной ОП (см. гл. 14).

В качестве ОП и СОП используются быстродействующие ЗУ с произвольным обращением и непосредственным доступом.

Понятие оперативной памяти выше определялось, исходя из выполняемых ею функций. Поскольку всегда в качестве ОП (как, впрочем, и СОП) используются ЗУ с произвольным обращением и непосредственным доступом, то часто последние независимо от выполняемых функций называют оперативными памятьми (оперативными ЗУ).

Обычно емкость ОП оказывается недостаточной для хранения всех необходимых данных в ЭВМ. Поэтому ЭВМ содержит в своем составе несколько ЗУ с прямым доступом на дисках (емкость одного ЗУ на дисках 10 — 300 Мбайт) и несколько ЗУ с последовательным доступом на магнитных лентах (емкость одного ЗУ 20—200 Мбайт).

Оперативная память вместе с СОП и некоторыми другими специализированными памятьми процессора образуют *внутреннюю память* ЭВМ (рис. 4.1). Электромеханические ЗУ образуют *внешнюю память* ЭВМ, а сами они поэтому называются *внешними запоминающими устройствами* (ВЗУ).

Современные ЭВМ содержат ряд специализированных быстродействующих памяти: памяти каналов, ключей защиты памяти, отдельных типов терминалов (дисплеев и др.), различные буферные памяти для промежуточного хранения информации при обмене ею между устройствами машины, работающими с различными скоростями. Наряду с этим используются также постоянные памяти, в основном для хранения микропрограмм, а в специализированных ЭВМ — и для хранения основных программ.

Запоминающее устройство любого типа состоит из *запоминающего массива*, хранящего информацию, и блоков, служащих для поиска в массиве, записи и считывания (а в ряде случаев и для регенерации) информации.

## **4.2. Адресная, ассоциативная и стековая организации памяти**

Запоминающее устройство с произвольным обращением, как правило, содержит множество одинаковых запоминающих элементов, образующих запоминающий массив (ЗМ). Массив раз-

делен на отдельные ячейки; каждая из них предназначена для хранения двоичного кода, число разрядов в котором определяется шириной выборки памяти (в частности, это может быть одно, половина или несколько машинных слов). Способ организации памяти зависит от методов размещения и поиска информации в запоминающем массиве. По этому признаку различают адресную, ассоциативную и стековую (магазинную) памяти.

**Адресная память.** В памяти с адресной организацией размещение и поиск информации в ЗМ основаны на использовании адреса хранения слова (числа, команды и т. п.). Адресом служит номер ячейки ЗМ, в которой это слово размещается.

При записи (или считывании) слова в ЗМ инициирующая эту операцию команда должна указывать адрес (номер ячейки), по которому производится запись (считывание).

Типичная структура адресной памяти, показанная на рис. 4.2, содержит запоминающий массив из  $N$   $n$ -разрядных ячеек и его аппаратное обрамление, включающее в себя регистр адреса  $PzA$ , имеющий  $k$  ( $k \geq \log N$ ) разрядов, информационный регистр  $PzH$ , блок адресной выборки  $БАВ$ , блок усилителей считывания  $БУС$ , блок разрядных усилителей-формирователей сигналов записи  $БУЗ$  и блок управления памятью  $БУП$ .

По коду адреса в  $PzA$   $БАВ$  формирует в соответствующей ячейке памяти сигналы, позволяющие произвести в ячейке считывание или запись слова.

Цикл обращения к памяти инициируется поступлением в  $БУП$  извне сигнала *Обращение*. Общая часть цикла обращения включает в себя прием в  $PzA$  с шины адреса  $ША$  адреса

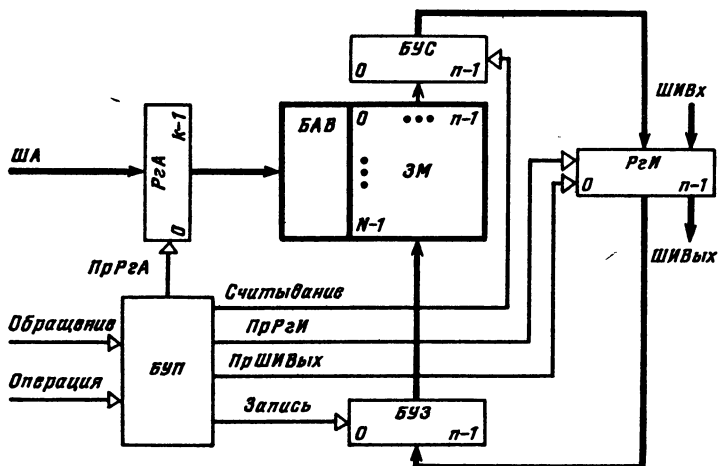


Рис. 4.2. Структура адресной памяти с произвольным обращением

обращения и прием в *БУП* и расшифровку управляющего сигнала *Операция*, указывающего вид запрашиваемой операции (считывание или запись).

Далее при считывании *БАВ* дешифрирует адрес, посылает сигналы считывания в заданную адресом ячейку *ЗМ*, при этом код записанного в ячейке слова считывается усилителями считывания *БУС* и передается в *РгИ*. Операция считывания завершается выдачей слова из *РгИ* на выходную информационную шину *ШИВых*.

При записи помимо выполнения указанной выше общей части цикла обращения производится прием записываемого слова с входной информационной шины *ШИВх* и *РгИ*. Затем в выбранную *БАВ* ячейку записывается слово из *РгИ*.

Блок управления *БУП* генерирует необходимые последовательности управляющих сигналов, инициирующих работу отдельных узлов памяти. Цепи передачи управляющих сигналов показаны тонкими линиями на рис. 4.2.

**Ассоциативная память.** В памяти этого типа поиск нужной информации производится не по адресу, а по ее содержанию (по ассоциативному признаку). При этом поиск по ассоциативному признаку (или последовательно по отдельным разрядам этого признака) происходит параллельно во времени для всех ячеек запоминающего массива. Во многих случаях ассоциативный поиск позволяет существенно упростить и ускорить обработку данных. Это достигается за счет того, что в памяти этого типа операция считывания информации совмещена с выполнением ряда логических операций.

Типичная структура ассоциативной памяти представлена на рис. 4.3. Запоминающий массив содержит  $N$  ( $n+1$ )-разрядных ячеек. Для указания занятости ячейки используется служебный  $n$ -й разряд (0 — ячейка свободна, 1 — в ячейке записано слово).

По входной информационной шине *ШИВх* в регистр ассоциативного признака *РгАП* в разряды  $0 \div n-1$  поступает  $n$ -разрядный ассоциативный запрос, а в регистр маски

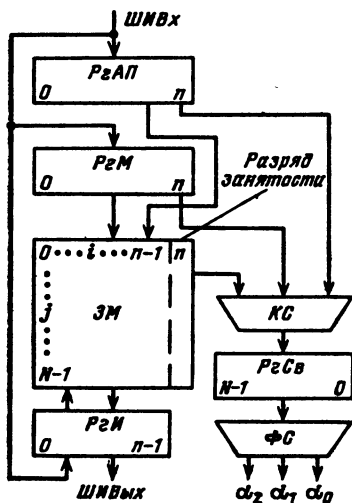


Рис. 4.3. Структура ассоциативной памяти

$P_2M$  — код маски поиска, при этом  $n$ -разряд  $P_2M$  устанавливается в 0. Ассоциативный поиск производится лишь для совокупности разрядов  $P_2АП$ , которым соответствуют 1 в  $P_2M$  (незамаскированные разряды  $P_2АП$ ). Для слов, в которых цифры в разрядах совпали с незамаскированными разрядами  $P_2АП$ , комбинационная схема  $КС$  устанавливает 1 в соответствующие разряды регистра совпадения  $P_2Св$  и 0 в остальные разряды. Таким образом, значение  $j$ -го разряда в  $P_2Св$  определяется выражением

$$P_2Св(j) = \bigwedge_{i=0}^{i=n-1} \{ \overline{P_2АП[i] \oplus 3М[j, i]} \vee \overline{P_2M[i]} \} \quad (0 \leq j \leq N-1),$$

где  $P_2АП[i]$ ,  $P_2M[i]$  и  $3М[j, i]$  — значения  $i$ -го разряда соответственно  $P_2АП$ ,  $P_2M$  и  $j$ -й ячейки  $3М$ .

Комбинационная схема формирования результата ассоциативного обращения  $\PhiС$  формирует из слова, образовавшегося в  $P_2Св$ , сигналы  $\alpha_0$ ,  $\alpha_1$ ,  $\alpha_2$ , соответствующие случаям отсутствия слов в  $3М$ , удовлетворяющих ассоциативному признаку, и наличия одного (и более) такого слова. Для этого  $\PhiС$  реализует следующие булевы функции:

$$\begin{aligned} \alpha_0 &= \bigwedge_{j=0}^{N-1} \overline{P_2Св[j]}, \\ \alpha_1 &= P_2Св[0] \overline{P_2Св[1]} \dots \overline{P_2Св[j]} \dots \overline{P_2Св[N-1]} \vee \\ &\vee \overline{P_2Св[0]} P_2Св[1] \overline{P_2Св[2]} \dots \overline{P_2Св[j]} \dots \overline{P_2Св[N-1]} \vee \dots \\ &\dots \vee \overline{P_2Св[0]} \overline{P_2Св[1]} \dots \overline{P_2Св[j]} \dots P_2Св[N-1]; \\ \alpha_2 &= \overline{\alpha_0} \alpha_1. \end{aligned}$$

Формирование содержимого  $P_2Св$  и сигналов  $\alpha_0$ ,  $\alpha_1$ ,  $\alpha_2$  по содержимому  $P_2АП$ ,  $P_2M$  и  $3М$  называется операцией контроля ассоциации. Эта операция является составной частью операций считывания и записи, хотя она имеет и самостоятельное значение.

При считывании сначала производится контроль ассоциации по ассоциативному признаку в  $P_2АП$ . Затем при  $\alpha_0 = 1$  считывание отменяется из-за отсутствия искомой информации, при  $\alpha = 1$  считывается в  $P_2И$  найденное слово, при  $\alpha_2 = 1$  в  $P_2И$  считывается слово из ячейки, имеющей наименьший номер среди ячеек, отмеченных 1 у  $P_2Св$ . Из  $P_2И$  считанное слово выдается на  $ШИВых$ .

При записи сначала отыскивается свободная ячейка. Для этого выполняется операция контроля ассоциации при  $P_2АП =$



$=111...10$  и  $P_2M=00...01$ , при этом свободные ячейки отмечаются 1 в  $P_2Cв$ . Для записи выбирается свободная ячейка с наименьшим номером. В нее записывается слово, поступившее с  $ШВХ$  в  $P_2И$ .

С помощью операции контроля ассоциации можно, не считывая слов из памяти, определить по содержимому  $P_2Cв$ , сколько в памяти слов, удовлетворяющих ассоциативному признаку, например реализовать запросы типа сколько студентов в группе имеют отличную оценку по данной дисциплине. При использовании соответствующих комбинационных схем в ассоциативной памяти могут выполняться достаточно сложные логические операции, такие, как поиск большего (меньшего) числа, поиск слов, заключенных в определенных границах, поиск максимального (минимального) числа и др.

Отметим, что для ассоциативной памяти необходимы запоминающие элементы, допускающие считывание без разрушения записанной в них информации. Это связано с тем, что при ассоциативном поиске считывание производится по всему  $ЗМ$  для всех незамаскированных разрядов и нигде сохранять временно разрушаемую считыванием информацию.

*Стековая память*, так же как и ассоциативная, является безадресной. Стековую память (рис. 4.4) можно рассматривать как совокупность ячеек, образующих одномерный массив, в котором соседние ячейки связаны друг с другом разрядными цепями передачи слов. Запись нового слова производится в верхнюю ячейку (ячейку 0), при этом все ранее записанные слова (включая слово, находившееся в ячейке 0), сдвигаются вниз, в соседние ячейки с большими на 1 номерами. Считывание возможно только из верхней (нулевой) ячейки памяти, при этом, если производится считывание с удалением, все остальные слова в памяти сдвигаются вверх, в соседние ячейки с большими номерами. В этой памяти порядок считывания слов соответствует правилу: *последним поступил — первым обслуживается*. В ряде устройств рассматриваемого типа предусматривается также операция простого считывания слова из нулевой ячейки (без его удаления и сдвига слова в памяти). Иногда стековая память снабжается счетчиком стека  $СчСт$ , показывающим количество

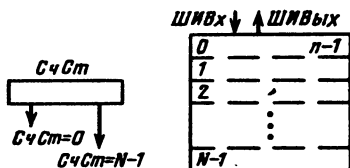


Рис. 4.4. Стековая память

занесенных в память слов. Сигнал  $CчСт=0$  соответствует пустому стеку,  $CчСт=N-1$  — заполненному стеку.

Обычно стековую память организуют, используя адресную память. В этом случае счетчик стека, как правило, отсутствует, так как количество слов в памяти можно выявить по указателю стека (см. гл. 8). Широкое применение стековая память находит при обработке вложенных структур данных.

В последующих параграфах описываются различные типы ЗУ с адресной организацией. В гл. 8 рассмотрено использование стековой памяти при выполнении безадресных команд и прерываний, а в гл. 14 — ассоциативной памяти в аппаратуре динамического распределения ОП.

### 4.3. Структуры адресных ЗУ

Техника оперативных памяти прошла большой путь развития. Большое значение имело появление в начале 50-х годов магнитных ЗУ, использующих в качестве запоминающего элемента (ЗЭ) ферритовые сердечники с прямоугольной петлей намагничивания. Магнитные ЗУ к настоящему времени вытеснены полупроводниковыми, в которых в качестве ЗЭ служат триггерные схемы или МОП-транзисторы.

Тип используемых ЗЭ определенным образом влияет на структуру памяти, в результате чего существует большое разнообразие структур ЗУ. В настоящем параграфе дается систематизация структур адресных ЗУ<sup>1</sup>, что должно помочь пониманию принципов действия различных типов ЗУ с произвольным обращением и постоянных ЗУ.

Совокупность определенным образом соединенных ЗЭ образует запоминающую матрицу или запоминающий массив, где каждый ЗЭ хранит бит информации. Запоминающий элемент должен реализовать следующие режимы работы: хранение состояния, выдача сигнала состояния (считывание), запись 0 или запись 1. К ЗЭ должны поступать управляющие сигналы для задания режима работы, а также информационный сигнал при записи, а при считывании ЗЭ должен выдавать сигнал о его состоянии.

Запоминающий массив имеет систему адресных и разрядных линий (проводников). Адресные линии используются для выделения по адресу совокупности ЗЭ, которым устанавливается режим считывания или записи. Выделение отдельных разрядов осуществляется разрядными линиями, по которым передается

---

<sup>1</sup> Систематизация структур ЗУ излагается в соответствии с [28].

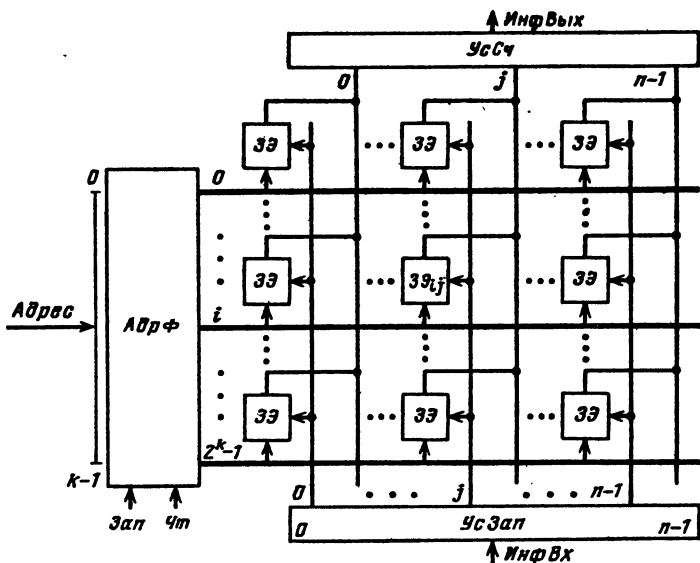


Рис. 4.5. Структура ЗУ типа 2D

записываемая в ЗЭ информация или информация о состоянии ЗЭ.

Запоминающие устройства строятся из специфичных ЗЭ, для которых характерно использование трончных сигналов и совмещение линий входных и выходных сигналов.

Адресные и разрядные линии носят общее название выборки. В зависимости от числа таких линий, соединенных с одним ЗЭ, различают двух- и трехкоординатные ЗУ и т. д., называемые ЗУ типа 2D, 3D и т. д. (от английского *dimension* — размерность). Наибольшее распространение получили ЗУ типа 2D, 3D, 2,5D и их модификации.

*Запоминающие устройства типа 2D.* Организация ЗУ типа 2D обеспечивает двухкоординатную выборку каждого ЗЭ ячейки памяти. Основу ЗУ составляет плоская матрица из ЗЭ, сгруппированных в  $2^k$  ячеек по  $n$  разрядов. Обращение к ячейке задается  $k$ -разрядным адресом, выделение разрядов производится разрядными линиями записи и считывания. Структура ЗУ типа 2D приведена на рис. 4.5.

Адрес ( $k$ -разрядный) выбираемой ячейки  $i$  поступает на схему адресного формирователя *АдрФ*, управляемого сигналами чтения *Чт* и записи *Зап*. Основу *АдрФ* составляет дешифратор с  $2^k$  выходами, который при поступлении на его входы адреса формирует сигнал для выборки линии  $i$ , при этом под воздей-

ствием сигналов *Чт* и *Зап* из *АдрФ* выдается сигнал, настраивающий ЗЭ *i*-й линии либо на считывание (выдачу сигнала состояния), либо на запись. Выделение разряда *j* в *i*-м слове производится второй координатной линией. При записи по линии *j* от усилителя записи *УсЗап* поступает сигнал, устанавливающий выбранный для записи ЗЭ<sub>*ij*</sub> в состояние 0 или 1. При считывании на усилитель считывания *УсСч* по линии *j* поступает сигнал о состоянии ЗЭ<sub>*ij*</sub>.

Используемые здесь ЗЭ должны допускать объединение выходов для работы на общую линию с передачей сигналов только от выбранного ЗЭ. Такое свойство типично для современных ЗЭ и в дальнейшем всякий раз подразумевается.

Таким образом, каждая адресная линия выборки ячейки передает три значения сигнала: выборка при записи, выборка при считывании и отсутствие выборки. Каждая разрядная линия записи передает в ЗЭ записываемый бит информации, а разрядная линия считывания — считываемый из ЗЭ бит информации. Линии записи и считывания могут быть объединены в одну при использовании ЗЭ, допускающих соединение выхода со входом записи. Совмещение функций записи и считывания на разрядной линии широко используется в современных полупроводниковых ЗУ.

Запоминающие устройства типа *2D* являются быстродействующими и достаточно удобными для реализации. Однако ЗУ типа *2D* неэкономичны по объему оборудования из-за наличия в них дешифратора с  $2^k$  выходами. В настоящее время структура типа *2D* используется в основном в ЗУ небольшой емкости.

*Запоминающие устройства типа 3D.* Некоторые ЗЭ имеют не один, а два конъюнктивных входа выборки. В этом случае адресная выборка осуществляется только при одновременном появлении двух сигналов. Использование таких ЗЭ позволяет строить ЗУ с трехкоординатным выделением ЗЭ.

Запоминающий массив ЗУ типа *3D* выполнен в виде пространственной матрицы, составленной из *n* плоских матриц, представляющих собой ЗМ для отдельных разрядов ячеек памяти. Запоминающие элементы для разряда сгруппированы в квадратную матрицу из  $\sqrt{2^k}$  рядов по  $\sqrt{2^k}$  ЗЭ в каждом.

Структура матрицы *j*-го разряда в ЗУ типа *3D* представлена на рис. 4.6. Для адресной выборки ЗЭ задаются две его координаты в *ЗМ<sub>j</sub>*. Код адреса *i*-й ячейки памяти разделяется на старшую и младшую части (*i'* и *i''*), каждая из которых поступает на свой адресный формирователь. Адресный формирователь *АдрФ1* выдает сигнал выборки на линию *i'*, а *АдрФ2* — на линию

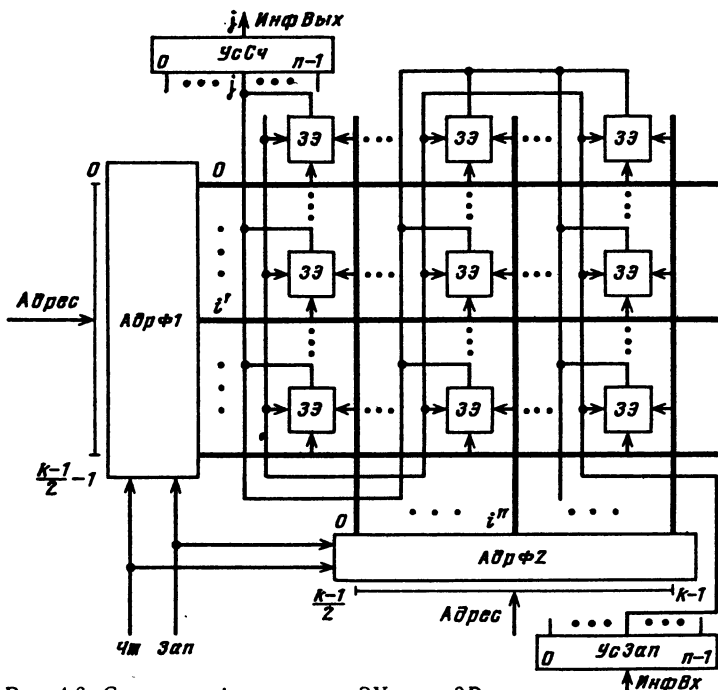


Рис. 4.6. Структура  $j$ -го разряда ЗУ типа 3D

$i''$ . В результате в  $ЗМj$  оказывается выбранным 3Э, находящийся на пересечении этих линий (двух координат), т. е. адресуемый кодом  $i = i' / i''$ . Адресные формирователи управляются сигналами  $Чт$  и  $Зап$  и в зависимости от них выдают сигналы выборки для считывания или записи. При считывании сигнал о состоянии выбранного 3Э поступает по  $j$ -й линии считывания к  $УсСч$  (третья координата 3Э). При записи в выбранный 3Э будут занесены 0 и 1 в зависимости от сигнала записи в  $j$ -й разряд, поступающего по  $j$ -й линии от  $УсЗап$  (третья координата 3Э при записи). Для полупроводниковых ЗУ, как отмечалось выше, характерно объединение в одну линию разрядных линий записи и считывания.

Для построения  $n$ -разрядной памяти используется  $n$  матриц рассмотренного вида. Адресные формирователи при этом могут быть общими для всех разрядных ЗМ.

Запоминающие устройства типа 3D более экономичны, чем ЗУ типа 2D. Действительно, сложность адресного формирователя с  $m$  входами пропорциональна  $2^m$ . Поэтому сложность двух адресных формирователей ЗУ типа 3D, пропорциональная

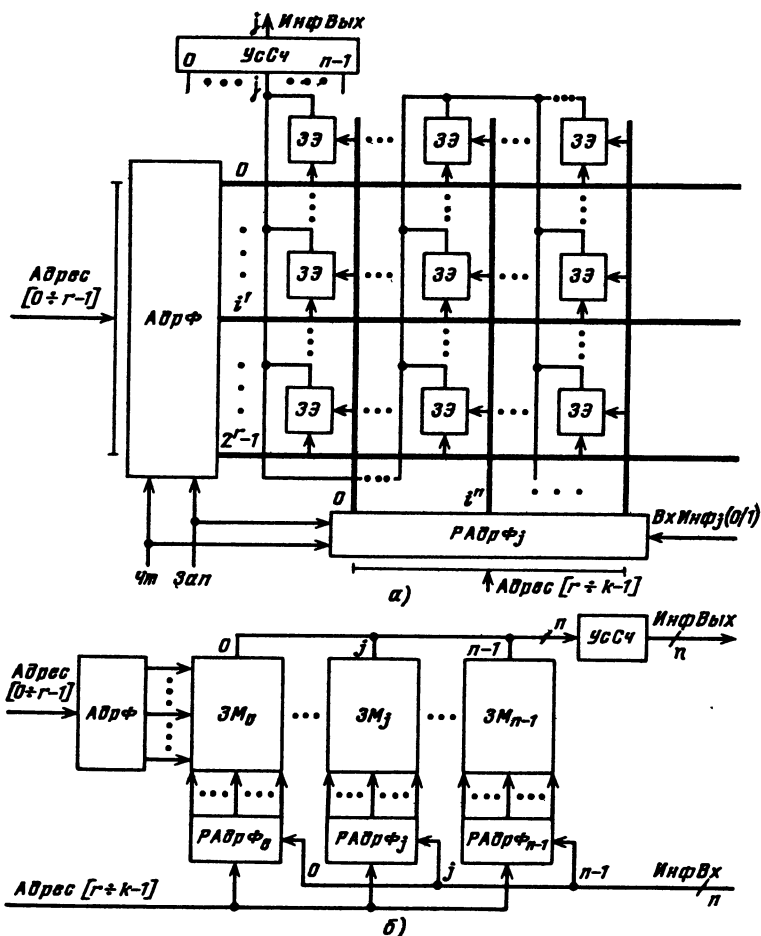


Рис. 4.7. Структура ЗУ типа 2,5D:  
а — одноразрядное ЗУ; б —  $n$ -разрядное ЗУ

$2 \cdot 2^{k/2}$ , значительно меньше сложности адресного формирователя ЗУ типа 2D, пропорциональной  $2^k$ . В связи с этим структура типа 3D позволяет строить ЗУ большего объема, чем структура 2D. Однако ЗЭ с тремя входами, используемыми при записи, не всегда удается реализовать.

**Запоминающие устройства типа 2,5D.** В ЗУ этого типа при считывании состояния  $j$ -го разряда  $i$ -й ячейки положение ЗЭ $_{ij}$  в ЗМ определяется тремя координатами (две координаты для

выборки и одна для выходного сигнала), а при записи в  $ЗЭ_{ij}$  — двумя координатами. Считывание при этом осуществляется так же, как и в  $ЗУ$  типа  $3D$ , а запись сходна с записью в  $ЗУ$  типа  $2D$ .

Запоминающий массив  $ЗУ$  типа  $2,5D$  можно рассматривать как состоящий из отдельных  $ЗМ$  для каждого разряда памяти:  $ЗМ_0, ЗМ_1, \dots, ЗМ_j, \dots, ЗМ_{n-1}$ . Структура одноразрядного  $ЗУ$  дана на рис. 4.7, а. Код адреса  $i$ -й ячейки памяти, как и в  $ЗУ 3D$ , разделяется на две части:  $i'$  и  $i''$ , каждая из которых отдельно дешифрируется. Адресный формирователь  $АдрФ$  выдает сигнал выборки на линию  $i'$ , разрядно-адресный формирователь  $j$ -го разряда  $РАдрФ$  — на линию  $i''$ . При считывании оба сигнала, являющиеся сигналами выборки для считывания, опрашивают  $ЗЭ$ , выходной сигнал которого поступает на  $УсСч$  разряда  $j$ . Работает  $ЗУ$  в этом случае так же, как и  $ЗУ$  типа  $3D$ .

При записи  $АдрФ$  выдает сигнал выборки для записи, а  $РАдрФ$  выдает по линии  $i''$  сигнал записи 0 или 1 в зависимости от назначения входного информационного сигнала  $j$ -го разряда  $ВхИнФ_j$ . На остальных линиях  $РАдрФ_j$  не появляются сигналы записи, и состояния всех  $ЗЭ$ , кроме  $ЗЭ$ , лежащего на пересечении линий  $i'$  и  $i''$ , не меняются.

Из  $ЗМ$  отдельных разрядов формируется  $ЗМ$  всего  $ЗУ$  согласно схеме на рис. 4.7, б.

Наиболее экономичным по расходу оборудования  $ЗУ$  оказывается в том случае, если число выходных линий  $АдрФ$  и всех  $РАдрФ$  равно, т. е. если  $r = (k - r) \log_2 n$  (рис. 4.7, б).

Недостатком  $ЗУ$  типа  $2,5D$  является то, что сигналы на линиях  $РАдрФ$  должны иметь четыре значения: чтение, запись 0, запись 1 и отсутствие записи (хранение). Для  $ЗЭ$  с разрушающим считыванием сигналы чтения и записи 0 совпадают и погребуются лишь три значения сигнала. В связи с этим  $ЗУ$  типа  $2,5D$  используется для  $ЗЭ$  с разрушающим считыванием.

Для построения современных полупроводниковых  $ЗУ$  из  $ЗЭ$  с неразрушающим считыванием используется структура  $ЗУ$  с двухкоординатным выделением  $ЗЭ$  и мультиплексированием выходных сигналов при считывании. Такие  $ЗУ$  будем называть  $ЗУ$  типа  $2D-M$ .

*Запоминающие устройства типа  $2D-M$ .* Запоминающие элементы таких  $ЗУ$  имеют два входа и один выход (рис. 4.8, а). При наличии хотя бы одного пустого сигнала  $\sim$  на входах  $ЗЭ$  при записи находится в режиме хранения (как в  $ЗУ 3D$ ). Сигнал чтения  $Чт$  опрашивает состояние  $ЗЭ$  (так же как и в  $ЗУ$  типа  $D$ ). Сигналы записи  $Зап$  и  $Зап 0$  устанавливают  $ЗЭ$  в состояние 0, а  $Зап$  и  $Зап 1$  — в состояние 1 (так же, как и в  $ЗУ$  типов  $2D$  и  $2,5D$ ).

Обычно у запоминающих элементов  $ЗУ$  типа  $2D-M$  выход

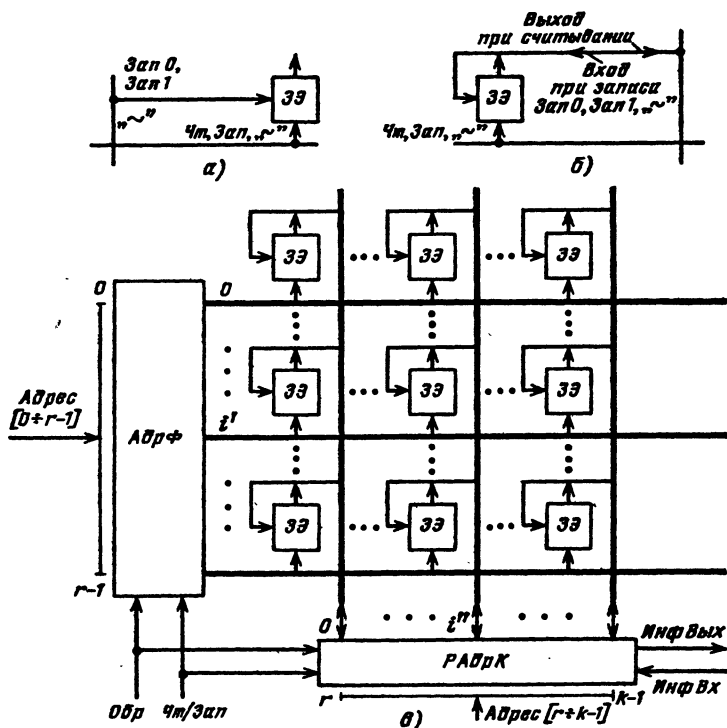


Рис. 4.8. Структура ЗУ типа 2D-M:

а — ЗЭ; б — ЗЭ с общим входом-выходом; в — структура одноразрядного ЗУ

объединяется со входом записи, как это показано для ЗЭ на рис. 4,8, б. Структура одноразрядного ЗУ типа 2D-M представлена на рис. 4,8, в. Как и в ЗУ типа 2,5 D, код адреса  $i$ -й ячейки разделяется на две части:  $i'$  и  $i''$ , одна из которых поступает на АдрФ, а другая — на разрядно-адресный коммутатор РАдрК. Если на АдрФ и РАдрК не приходит сигнал обращения к памяти Обр, то на их выходных линиях не возникают действующие на ЗЭ сигналы и все ЗЭ находятся в режиме хранения. При наличии сигнала Обр выполняется считывание или запись в зависимости от значения сигнала Чт/Зап. При считывании АдрФ выдает по линии  $i'$  сигнал выборки для считывания, по которому со всех ЗЭ линии  $i'$  сигналы их состояний поступают на РАдрК. Коммутатор РАдрК мультиплексирует эти сигналы и передает на выход ИнфВых сигнал с линии  $i''$ . При записи АдрФ выдает по линии  $i'$  сигнал выборки для записи. Коммутатор РАдрК в зависимости от значения ИнфВх выдает сигнал записи 0 или 1 на линию  $i''$  и сигналы, не воздействующие на ЗЭ, в остальные



линии. В результате запись производится только в ЗЭ, лежащий на пересечении координатных линий  $i'$  и  $i''$ , причем  $i'/i'' = i$ .

Построив схему, аналогичную схеме на рис. 4.7, б, получим ЗУ для  $2^k n$ -разрядных ячеек. Наиболее экономична такая схема при  $r = (k - r) \log_2 n$ .

Структура типа 2D-M наиболее удобна для построения полупроводниковых ЗУ и широко используется в настоящее время как в оперативных, так и в постоянных ЗУ.

#### 4.4. Запоминающие устройства с произвольным обращением

В вычислительной технике в качестве ЗУ с произвольным обращением, используемых в оперативных памяти ЭВМ, еще недавно широко применялись ЗУ с ЗЭ на ферритовых сердечниках. Успехи в технологии БИС привели к созданию *полупроводниковых интегральных ЗУ*, на основе которых создаются основные (оперативные) памяти современных ЭВМ.

По сравнению с ферритовыми ЗУ полупроводниковые имеют ряд важных достоинств: большее быстродействие, компактность, меньшую стоимость, совместимость по сигналам с логическими схемами, общие с другими электронными устройствами ЭВМ технологические и конструктивные принципы построения.

Недостатком полупроводниковых ЗУ с произвольным обращением является их энергозависимость, выражающаяся в том, что они потребляют энергию в режиме хранения информации и теряют информацию при выключении напряжения питания (потери информации можно избежать автоматическим переключением на аварийное питание от аккумуляторов).

По типу ЗЭ различают *биполярные ЗУ* с биполярными транзисторами (с ТТЛ- или ЭСЛ-схемами) и МОП-ЗУ с МОП-транзисторами.

**Биполярные ЗУ.** В биполярных интегральных ЗУ в качестве ЗЭ используется статический триггер на двух многоэмиттерных транзисторах с непосредственными связями (рис. 4.9).

Эмиттеры 11 и 21 являются парафазными информационными входами ЗЭ и служат для записи в триггер 1 или 0. Эти же эмиттеры используются как выходы при считывании информации. Адресные эмиттеры 12, 22, 13 и 23 образуют два конъюнктивно связанных входа выборки.

Организация ЗУ из триггеров осуществляется по схеме типа 3D.

В режиме хранения (ЗЭ не выбран) эмиттерный ток открытого транзистора замыкается на землю через адресные эмиттеры и адресные линии (или только через один такой эмиттер и одну линию), находящиеся под потенциалом логического 0 ( $\leq 0,4 В$ ). При этом информацион-

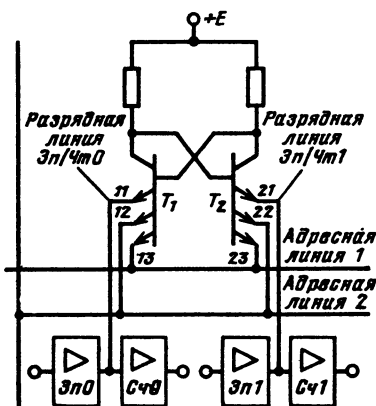


Рис. 4.9. Запоминающий элемент полупроводникового биполярного ЗУ

ные эмиттеры должны быть заперты, для чего на них подается потенциал (1—1,5 В), который больше потенциала адресных эмиттеров (больше максимального значения уровня сигнала логического 0, равного 0,4 В, но меньше минимального значения сигнала логической 1, составляющего 2,4 В), с тем чтобы при выборке ЗЭ через информационные эмиттеры протекали токи, необходимые для операций считывания и записи.

При выборке данного ЗЭ на его адресные эмиттеры с выходов адресных дешифраторов подается потенциал логической 1 ( $\geq 2,4$  В), превышающий потенциал информационных эмиттеров. Поэтому адресные эмиттеры оказываются запертыми, а коллекторный ток открытого транзистора течет через его информационный эмиттер, чем обеспечивается возможность считывания из ЗЭ и записи в него информации.

Состояния 1 и 0 ЗЭ распознаются по наличию тока соответственно в разрядной линии 0 (открыт транзистор  $T_1$ ) или в разрядной линии 1 (открыт транзистор  $T_2$ ).

Считывание происходит без разрушения информации. Хранимая в ЗЭ информация доступна для считывания все время, пока ЗЭ находится в выбранном состоянии, и в него не производится запись (отсутствует импульс «разрешение записи»).

При считывании на входы обоих усилителей записи подается потенциал логического 0, в результате чего на выходах этих усилителей оказывается потенциал логической 1, запирающий усилители записи и тем самым предотвращающий ответвление в них тока считывания (тока информационного эмиттера).

При считывании ток вытекает из информационного эмиттера открытого транзистора и втекает в базовую цепь входного транзистора соответствующего усилителя считывания, в результате чего выходной транзистор последнего полностью открывается.

Для записи в ЗЭ 1 или 0 с соответствующего усилителя записи на подключенный к нему информационный эмиттер подается потенциал логического 0 ( $\leq 0,4$  В), а на другой информационный эмиттер продолжает поступать с его невозбужденного усилителя записи потенциал, равный примерно 1,5 В.

Если допустим, производится запись 1 в триггер, находившийся перед этим в состоянии 1 (открыт транзистор  $T_2$ ), то подача потенциала низкого уровня на эмиттер 21 не меняет состояние триггера. Если до записи триггер находился в состоянии 0, то при подаче потенциала низкого уровня на эмиттер 21 (запись 1) открывается транзистор  $T_2$ , при этом транзистор  $T_1$  закрывается и триггер устанавливается в состояние 1.

Интегральная микросхема биполярного ЗУ представляет собой кристалл кремния, в котором образованы массив ЗЭ (триггеров) со всеми межсоединениями, а также адресные дешифраторы, усилители-формирователи записи и считывания и другие схемы для управления адресной выборкой, записью и считыванием. Для повышения быстродействия ЗУ эти обслуживающие схемы могут быть выполнены на основе ЭСЛ-элементов, работающих в линейной области, в то время как построенные на основе ТТЛ-элементов триггеры ЗЭ работают с насыщением. В таком случае кристалл содержит схемы согласования уровней сигналов для перехода от схем ТТЛ к схемам ЭСЛ и обратно.

Полупроводниковые ЗУ размещаются в стандартных корпусах интегральных микросхем. Число выводов ограничивают число слов и разрядов запоминающего массива интегральной микросхемы. Для получения ЗУ с большим числом разрядов и (или) слов, чем в запоминающем массиве в корпусе схемы, применяются несколько корпусов.

В настоящее время биполярные ЗУ довольно дороги, поэтому они используются главным образом в качестве сверхоперативных памяти.

В динамических ЗУ двоичные коды хранятся на «запоминающих емкостях», в качестве которых используются паразитные емкости некоторых цепей схем. Примем, что отсутствие заряда на запоминающей емкости означает состояние 0, а наличие — состояние 1. В таком случае считывание информации состоит в определении, заряжены или нет запоминающие емкости.

Схема и временные диаграммы работ ЗЭ динамического ЗУ на МОП-транзисторах в памяти со структурой  $2D-M$  представлены на рис. 4.10. Запоминающей емкостью служит паразитная емкость  $C$  затвора транзистора  $T_2$ . Линия разрядно-адресного коммутатора  $Y$  используется для ввода в ЗЭ бита информации при записи и съема его при считывании (см. рис. 4.8). Так как ЗЭ использует источник питания только при считывании, то им может служить паразитная емкость  $C_Y$  линии  $Y$ .

Предварительно перед считыванием от разрядно-адресного коммутатора подается сигнал  $R$ , с помощью которого подготавливается считывание с мультимплексированием для ЗЭ, выбираемых линией разрядно-адресного формирователя. Сигнал  $R$  открывает транзистор  $T_4$ , и емкость  $C_Y$  подзаряжается от источника. Затем на линию  $X$  подается от адресного формирователя сигнал считывания — промежуточный уровень сигнала  $CWR$ , который открывает транзистор  $T_3$ , но не может открыть  $T_2$ . Если ЗЭ хранит 1, то конденсатор  $C$  заряжен и открыт транзистор  $T_2$ . В этом случае через открытые транзисторы  $T_3$  и  $T_2$  конденсатор  $C_Y$  разряжается

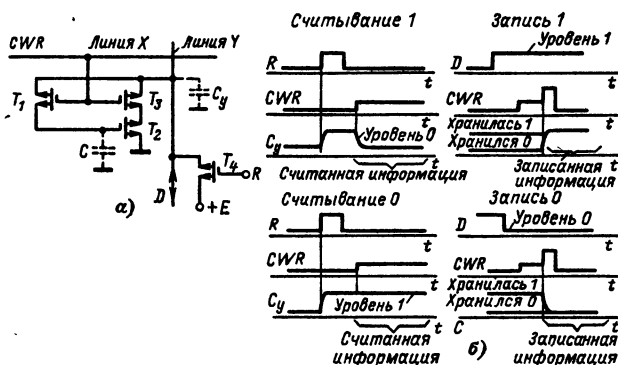


Рис. 4.10. Принципиальная электрическая схема (а) и временные диаграммы работы (б) ЗЭ динамического МОП-ЗУ

**Таблица 4.1. Параметры и области применения БИС ЗУ с произвольным обращением**

Тип ЗУ	Емкость, Кбит	Время выборки, нс	Область применения			
			ОП больших ЭВМ	ОП средних и небольших ЭВМ	ОП микро-ЭВМ	Сверхоперативная и буферная памяти
Статическое ЭСЛ	1—4	10—20				+
Статическое ТГЛ	1—4	25—50				+
Статическое <i>n</i> -МОП	16—64	50—150		+	+	
Статическое КМОП	16—64	50—150		+	+	
Динамическое <i>n</i> -МОП	64—256	120—150	+	+	+	
Динамическое КМОП	256—1024	80—120	+	+	+	

и низкий уровень (уровень 0) сигнала *D* на линии *Y* указывает, что ЗЭ хранил инверсное значение, т. е. 1. Если ЗЭ хранит 0, то емкость *C* разряжена, *T<sub>2</sub>* закрыт и сигнал *CWR* не может вызвать разряд емкости *C<sub>γ</sub>*. Высокий уровень сигнала *D* (уровень 1) указывает, что ЗЭ хранил 0. Далее сигнал *D* через разрядно-адресный коммутатор поступает на выход ЗУ.

При записи на линию *Y* поступает сигнал *D*, соответствующий записываемому двоичному знаку. Затем на линию *X* подается высокий уровень сигнала *CWR*, открывающий транзистор *T<sub>1</sub>*, который подключает к линии *Y* конденсатор *C*. В результате независимо от своего предыдущего состояния емкость оказывается заряженной, если записывается 1, и разряженной, если записывается 0.

В ЗУ периодически производится регенерация информации. При регенерации в ЗЭ записывается инверсное значение хранимого до считывания кода. После каждой четной регенерации — его инверсия. В ЗУ имеется схема, сигнал которой указывает, какой код хранить в данный момент ЗЭ — прямой или инверсный.

В настоящее время большие оперативные памяти ЭВМ выполняют главным образом на динамических МОП-ЗУ, небольшие ОП — на МОП-ЗУ и ТТЛ-ЗУ, а сверхоперативные и буферные памяти — на ЭСЛ-ЗУ и ТТЛ-ЗУ.

В табл. 4.1 приведены характерные параметры БИС для разных типов полупроводниковых ЗУ и указаны области их использования.

## 4.5. Постоянные ЗУ

*Постоянные запоминающие устройства* (ПЗУ) в рабочем режиме ЭВМ допускают только считывание хранимой информации. В зависимости от типа ПЗУ занесение в него информации производится в процессе

или изготовления, или эксплуатации путем настройки, предваряющей использование ПЗУ в вычислительном процессе. В последнем случае ПЗУ называется постоянным запоминающим устройством с изменяемым в процессе эксплуатации содержимым или программируемыми постоянными запоминающими устройствами (ППЗУ).

Постоянные запоминающие устройства обычно строятся как адресные ЗУ. Функционирование ПЗУ можно рассматривать как выполнение однозначного преобразования  $k$ -разрядного кода адреса ячейки запоминающего массива (ЗМ) в  $n$ -разрядный код хранящегося в ней слова. При такой точке зрения ПЗУ можно считать преобразователем кодов или комбинационной схемой (автоматом без памяти) с  $k$  входами и  $n$  выходами.

По сравнению с ЗУ с произвольным обращением, допускающим как считывание, так и запись информации, конструкции ПЗУ значительно проще, их быстродействие и надежность выше, а стоимость ниже. Это объясняется большей простотой ЗЭ, отсутствием цепей для записи информации вообще или по крайней мере для оперативной записи, реализацией неразрушающего считывания.

Одним из важнейших применений ПЗУ является хранение микропрограмм в микропрограммных управляющих устройствах ЭВМ. Для этой цели необходимы ПЗУ значительно большего, чем в ОП, быстродействия и умеренной емкости (10 000—100 000 бит).

Постоянные запоминающие устройства широко используются для хранения программ в специализированных ЭВМ, в том числе в микро-ЭВМ, предназначенных для решения определенного набора задач, для которых имеются отработанные алгоритмы и программы, например в бортовых ЭВМ самолетов, ракет и космических кораблей, в управляющих вычислительных комплексах, работающих в АСУ технологическими процессами. Такое применение ПЗУ позволяет существенно снизить требования к емкости ОП, повысить надежность и уменьшить стоимость вычислительной установки.

На рис. 4.11, а приведена схема простейшего ПЗУ со структурой типа 2D. Запоминающий массив образуется системой взаимно перпендикулярных линий, в их пересечениях устанавливаются ЗЭ, которые либо связывают (состояние 1), либо не связывают (состояние 0) между собой соответствующие горизонтальную и вертикальную линии. Поэтому часто ЗЭ и ПЗУ называют связывающими элементами. Для некоторых типов

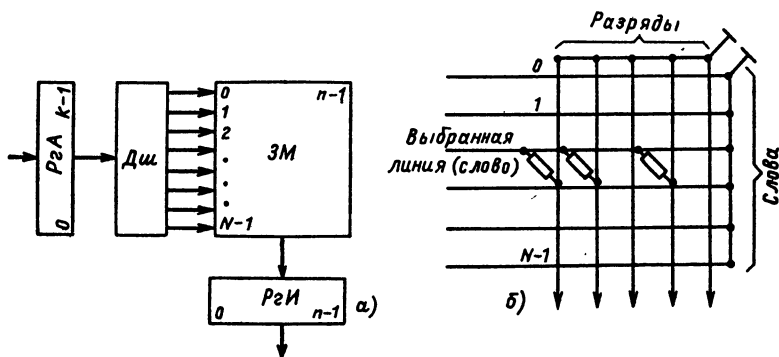


Рис. 4.11. Постоянное ЗУ типа 2D

ЗЭ состояние 0 означает просто отсутствие запоминающего (связывающего) элемента в данной позиции в ЗМ.

Дешифратор  $Dш$  по коду адреса в  $P_2A$  выбирает одну из горизонтальных линий (одну из ячеек ЭВМ), в которую подается сигнал выборки. Выходной сигнал 1 появляется на тех вертикальных разрядных линиях, которые имеют связь с возбужденной адресной линией (на рис. 4.11, б считывается слово 11010).

В зависимости от типа запоминающих (связывающих) элементов различают резисторные, емкостные, индуктивные (трансформаторные), полупроводниковые (интегральные) и другие ПЗУ.

В настоящее время наиболее распространенным типом являются полупроводниковые интегральные ПЗУ.

**Полупроводниковые интегральные ПЗУ.** Полупроводниковые ПЗУ имеют все те же достоинства, которые отмечались в предыдущем параграфе в отношении полупроводниковых ЗУ с произвольным обращением. Более того, в отличие от последних они являются энергонезависимыми. Постоянные ЗУ имеют большую емкость на одном кристалле (в одном корпусе интегральной микросхемы).

Положительным свойством интегральных ПЗУ является то, что некоторые типы этих устройств позволяют самому потребителю производить их программирование (занесение информации) в условиях эксплуатации и даже многократное перепрограммирование.

По типу ЗЭ, устанавливающих или разрывающих связь (контакт) между горизонтальными и вертикальными линиями, различают биполярные и МОП-схемы ПЗУ. Биполярные ПЗУ имеют время выборки 30—50 нс и емкость в одном кристалле (корпусе) от 256 бит до 16 Кбит. Постоянные маскируемые ЗУ на МОП-схемах имеют большую емкость в одном кристалле (корпусе) — до 64 Кбит и более, но и значительно меньшее быстродействие: время выборки 100—200 нс.

По важнейшему признаку — способу занесения информации — реализуют три типа интегральных полупроводниковых ПЗУ: 1) с программированием в процессе изготовления путем нанесения при помощи фотошаблонов в нужных потребителю точках контактных перемычек; 2) с программированием выжиганием перемычек или пробоем  $p$ - $n$ -переходов, с помощью которых сам потребитель уже после изготовления прибора может уничтожить или образовать связи между горизонтальными и вертикальными линиями ЗМ (одноразовое программирование); 3)

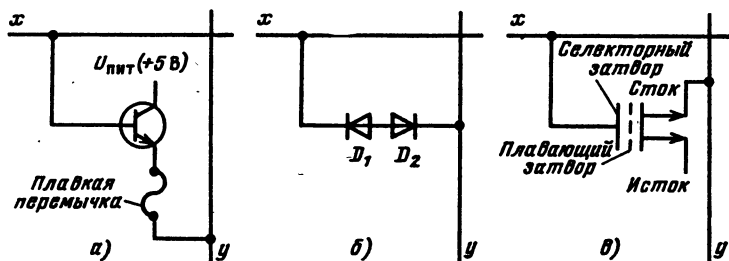


Рис. 4.12. Запоминающие (связывающие) элементы программируемых интегральных полупроводниковых постоянных ЗУ:

а — элемент с плавающей (выжигаемой) перемычкой; б — элемент с пробиваемым  $p$ - $n$ -переходом; в — лавинно-инжекционный МОП-транзистор с плавающим и селективным затворами

с электрическим перепрограммированием, при котором информация заносится в ЗМ электрическим путем, а стирание информации, необходимое для изменения содержимого ПЗУ, выполняется воздействием на ЗМ ультрафиолетового излучения или электрическим путем (многократное программирование).

Программируемые фотошаблонами и выжиганием ПЗУ могут строиться на основе как биполярных, так и МОП-схем. Перепрограммируемые ПЗУ используют только МОП-схемы, способные хранить заряды.

Различные типы ЗЭ интегральных ПЗУ представлены на рис. 4.12. На рис. 4.12, *а* показан биполярный транзисторный ЗЭ с выжигаемой перемычкой, соединяющий горизонтальную и вертикальную линии. При «программировании» ПЗУ перемычки выжигаются в нужных местах серийной импульсов тока с амплитудой 20—30 мА. При выборе адресным дешифратором горизонтальной линии  $x$  на базу транзистора ЗЭ поступает открывающий его сигнал, и при наличии перемычки (состояние 1) на вертикальной линии  $y$  появится потенциал коллектора транзистора  $+5$  В.

На рис. 4.12, *б* изображен ЗЭ, программируемый пробиванием  $p$ - $n$ -перехода. В исходном состоянии включенные встречно диоды изолируют линии  $x$  и  $y$  (состояние 0). При подаче повышенного напряжения диод  $D_2$  пробивается и закорачивается (состояние 1).

Более просто устроены ПЗУ с транзисторными и диодными запинающими (связывающими) элементами, программируемые при изготовлении ПЗУ. В этом случае с помощью фотошаблонов в нужных позициях ЗМ наносятся или не наносятся контактные перемычки (вместо плавкой перемычки и вместо диода  $D_1$  на рис. 4.12, *а* и *б* соответственно).

На рис. 4.12, *в* представлен ЗЭ в виде лавинно-инжекционного МОП-транзистора с плавающим и селектирующим затворами. Интегральные ПЗУ на таких элементах допускают многократную замену хранимой информации.

Плавающий (изолированный) затвор не имеет электрического подвода, он предназначен для хранения заряда. Селектирующий затвор подсоединен к одному из выходов дешифратора строк — горизонтальной линии, а сток — к вертикальной линии. В исходном состоянии отсутствует заряд на плавающем затворе (состояние 1), транзистор имеет очень небольшое пороговое напряжение. Выбор элемента осуществляется путем подачи на селектирующий затвор выходного напряжения адресного дешифратора, при этом включается транзистор и через цепь сток — исток протекает значительный ток. Программирование (занесение 0 в элементы) производится подачей на сток импульса напряжения 25—50 В, при этом происходит инжекция электронов, имеющих высокую энергию, через оксид на изолированный затвор, получающий отрицательный заряд (состояние 0). В результате увеличивается пороговое напряжение, и подача на селектирующий затвор выходного напряжения дешифратора не включает этот транзистор.

Структура программируемого ПЗУ емкость 8 К ( $1\text{ К} \times 8$ ) бит на одном кристалле (в одном корпусе) изображена на рис. 4.13.

Рассматриваемое перепрограммируемое ПЗУ имеет структуру типа 2D-M. Запоминающий массив содержит 64 горизонтальных  $X$  и 128 вертикальных  $Y$  линий, на пересечении которых расположены МОП-транзисторы с плавающим и селектирующим затворами. Вертикальные линии разбиты на 8 групп по 16 в каждой. Число групп соответствует числу разрядов, хранимых в микросхеме (корпусе) слов. В качестве адресного формирователя используется дешифратор линий  $X$ , выдающий сигналы чтения. Разрядно-адресный коммутатор образован дешифратором линий  $Y$ , который управляет коммутирующими транзисторами, подсоединенными к разрядным усилителям считывания-записи.

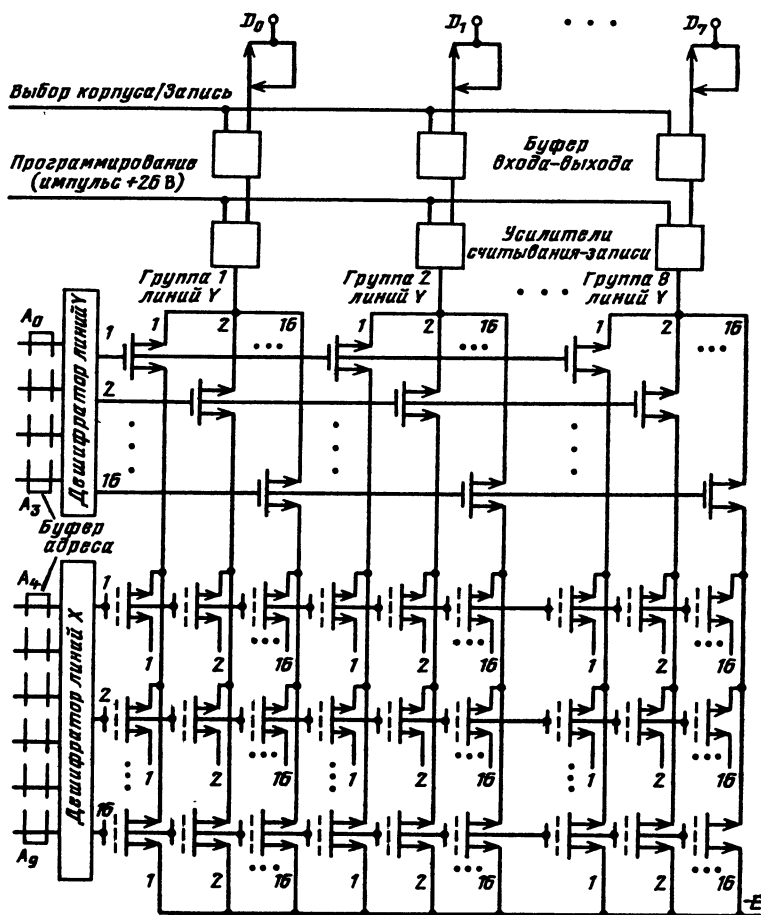


Рис. 4.13. Перепрограммируемое МОП-ПЗУ емкостью 8 К (1 К×8) бит (в одном корпусе)

При подаче младших разрядов адреса  $A_4—A_7$  на дешифратор линий  $X$  выбирается одна из 64 горизонтальных линий, в результате чего управляющий потенциал поступает на присоединенные к ней линии, селектирующие затворы 128 ЗЭ. В каждой группе линий по коду старших разрядов адреса  $A_0—A_3$  мультиплексируются сигналы от выбранных ЗЭ и выделяется сигнал адресуемого ЗЭ. Разрядные усилители формируют его и через буфера входа-выхода выдают на выходы  $D_0—D_7$  микросхемы сигналы логических 0 и 1, соответствующие информации, находящейся в выбранной ячейке. При считывании информация в ячейке сохраняется.



Рассматриваемое ПЗУ может работать в двух режимах: 1) считывание хранимой информации и 2) программирование (запись) ПЗУ. Режим считывания устанавливается подачей сигнала выбора корпуса, который настраивает схемы буферов входа-выхода и разрядных усилителей на формирование и передачу на выходы  $D_0—D_7$  сигналов, считанных с ЗЭ.

Перед записью производится стирание информации воздействием ультрафиолетового света через окно в корпусе микросхемы на полупроводниковый кристалл ПЗУ. Под воздействием света заряды стекают с затворов. После стирания все ЗЭ находятся в состоянии 1.

Режим записи осуществляется при одновременной подаче сигнала *Запись* и импульса *Программирование*. На входы корпуса  $A_0—A_9$  поступает адрес ячейки, в которую производится запись, а на выходы  $D_0—D_7$  (в режиме записи ставшие входами) записываемое 8-разрядное слово. Сигнал *Запись* настраивает буфера входа-выхода и разрядные усилители на передачу информационных сигналов со входов  $D_0—D_7$  к стокам ЗЭ, при этом открываются усилители записи тех разрядов, в которых в записываемом слове находятся 0. Эти усилители пропускают в выбранные дешифратором линий  $Y$  вертикальные линии импульс *Программирование* (импульс  $+26$  В), который обеспечивает лавинный пробой и занесение отрицательного заряда в плавающие затворы (запись 0) только тех ЗЭ, селектирующие затворы которых присоединены к горизонтальной линии, возбужденной выходным сигналом дешифратора линий  $X$ .

Рассматриваемое устройство имеет время выборки 0,4—1 мкс.

### Контрольные вопросы

1. Какие типы обращения и доступа характеризуют ЗУ оперативной (основной) памяти?
2. Как происходит поиск информации в ассоциативной памяти?
3. Дайте сравнительную характеристику различных структур адресных ЗУ.
4. Дайте сравнительную характеристику различных типов полупроводниковых ЗУ с произвольным обращением.
5. Укажите на рис. 4.13 транзисторы, хранящие информацию нулевого адреса ПЗУ.

## Глава 5

# ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА ЭВМ: ВНЕШНИЕ ЗУ И УСТРОЙСТВА ВВОДА-ВЫВОДА

## 5.1. Общие понятия о периферийных устройствах

В состав современных ЭВМ и систем входят многочисленные и разнообразные по выполняемым функциям, принципам действия и характеристикам *периферийные устройства* (ПУ), которые по их назначению можно разделить на две группы: 1) *внешние*

*запоминающие устройства*, предназначенные для хранения больших объемов информации, и 2) *устройства ввода-вывода*, обеспечивающие связь машины с внешней средой, в первую очередь с пользователями, путем ввода в ЭВМ и вывода из машины информации, ее регистрации и отображения.

*Устройства ввода* позволяют вводить в машину данные и программы, а также вносить исправления в программы и данные, хранящиеся в памяти ЭВМ. *Устройства вывода* служат для вывода из ЭВМ результатов обработки данных, их регистрации и отображения.

Общей характеристикой для всех периферийных устройств является скорость, с которой устройство может принимать или выдавать данные. Большинство периферийных устройств имеют электромеханические узлы, скорость работы которых значительно ниже скорости работы электронных устройств ЭВМ (процессоров и др.). Скорости передачи данных, с которыми работают различные ПУ, отличаются весьма значительно: от нескольких байт до нескольких миллионов байт в секунду.

Периферийные устройства различают по реализуемому в них синхронному или асинхронному режиму передачи (приема) данных.

При синхронном режиме передача данных производится в определенном темпе, который задается рабочей скоростью движения носителя информации, например магнитной ленты. При *асинхронном режиме* передача данных может происходить в свободном темпе с остановом после передачи любого байта.

В *устройствах ручного ввода* (пульта управления ЭВМ, электрифицированные пишущие машинки, дисплей и др.) ввод информации осуществляется с помощью клавиатур. Устройства ручного ввода позволяют вводить информацию со сравнительно низкой скоростью [не более 10 символов (знаков)/с].

В группу *автоматических устройств ввода* входят устройства для считывания информации с промежуточного носителя и устройства непосредственного ввода.

К устройствам ввода с промежуточного носителя информации относятся устройства считывания информации с перфокарт или перфолент, а также с магнитных лент. Информация наносится на перфокарты и перфоленты с помощью пробивок отверстий (перфораций). Устройства ввода с перфолент и перфокарт сравнительно просты и имеют относительно устройств ручного ввода высокие скорости ввода информации. Однако они требуют предварительной ручной подготовки данных — перфорирования перфокарт и перфолент.

Перфокарты и перфоленты как носители информации имеют свои преимущества и недостатки. Перфокарты более долговеч-

ны, позволяют легко сменить часть информации (путем простой замены некоторых карт), удобны для хранения, но они дороже перфолент. Перфокартное оборудование, используемое при подготовке перфокарт, вводе и выводе информации, более громоздко и более сложно в эксплуатации и имеет стоимость, большую стоимости оборудования для перфолент.

Для подготовки данных и ввода их в ЭВМ используются также устройства с магнитной лентой.

К автоматическим устройствам непосредственного ввода информации относятся устройства, считывающие информацию со специальных бланков и с графиков. Хотя подобные устройства достаточно сложны и дороги, их применение весьма целесообразно, так как позволяет исключить ручную подготовку данных (перфорация и т. п.), требующую больших затрат труда и нередко сопровождающуюся ошибками. В настоящее время ведутся интенсивные разработки устройств ввода с печатного текста, сделаны первые шаги по практическому использованию устройства ввода информации с голоса.

К автоматическим устройствам непосредственного ввода информации относятся также преобразователи аналоговых сигналов в цифровые, устройства приема информации с линией связи и др.

*Устройства вывода* информации из ЭВМ можно разделить на следующие группы:

1) устройства вывода цифровой информации на промежуточный носитель (перфокарты, перфоленты, магнитные ленты);

2) устройства, фиксирующие результаты обработки в виде текста, графиков и изображений, — различного рода печатающие устройства, устройства вывода информации на разного рода экраны, графопостроители;

3) устройства вывода информации во внешнюю среду (цифро-аналоговые преобразователи и устройства выдачи данных на линии связи);

4) устройства речевого вывода.

В настоящей главе после описания общих принципов построения и функционирования ЗУ на магнитных лентах и дисках рассмотрены особенности комплекса периферийных устройств современных персональных компьютеров, с которыми пользователю ПК приходится иметь дело.

С другими типами периферийных устройств, в том числе характерных для машин общего назначения (устройства почтовой печати и др.), читатель может ознакомиться в [21], а с аналого-цифровыми и цифро-аналоговыми преобразователями — в [24].

## 5.2. Принципы действия внешних ЗУ

Иерархически организуемая память ЭВМ (см. § 4.1) наряду с основной (оперативной) памятью высокого быстродействия (цикл обращения 1 мкс и менее), но сравнительно небольшой емкости (в больших ЭВМ — до нескольких мегабайт) содержит внешнюю память, намного медленнее работающую, но способную хранить практически сколь угодно большое количество необходимой для функционирования вычислительной установки информации (данные, программы и др.).

*Внешняя память* состоит из нескольких внешних запоминающих устройств (ВЗУ), в качестве которых в современных ЭВМ используются, главным образом, *электрохимические ЗУ* с носителем информации в виде движущейся поверхности, покрытой тонким слоем магнитного материала. Внешние ЗУ являются устройствами с произвольным обращением, допускающими многократное считывание информации и запись новой информации на место ранее записанной. Электрохимические ВЗУ компактны, сравнительно дешевы (в пересчете стоимости на 1 бит хранимой информации), могут хранить в одном устройстве (модуле) сотни миллионов байт. Необходимая емкость внешней памяти достигается подсоединением к ЭВМ соответствующего количества ВЗУ.

В вычислительной технике используются электрохимические ВЗУ с носителем информации в виде магнитных лент, дисков и, реже, барабанов.

Основу процесса магнитной записи составляет взаимодействие магнитного носителя информации и магнитных головок.

Магнитные головки представляют собой миниатюрные электромагниты, располагаемые у поверхности движущегося магнитного носителя с небольшим зазором при *бесконтактной записи* или без зазора при *контактной записи*. Информация записывается на носителе вдоль дорожки, проходящей под головкой.

Сердечники головки изготавливаются из материала с малой коэрцитивной силой и большим значением индукции насыщения. Наоборот, магнитный материал носителя должен обладать достаточной коэрцитивной силой, чтобы записанная на носителе информация длительно сохранялась и не стиралась под действием внешнего поля. Значение коэрцитивной силы для материала магнитного слоя лент лежит в пределах 12 000—24 000 А/м, а для дисков и барабанов 24 000—80 000 А/м. Остаточная индукция для лент обычно составляет 0,08—0,15 Тл, а для дисков и барабанов — от 0,15 до 0,6 Тл.

Для запоминания информации используют два противоположных состояния насыщения магнитного материала носителя.

При записи в обмотку магнитной головки посылается соответствующая записываемой информации и используемому методу записи последовательность положительных и отрицательных (или только однополярных) потенциальных сигналов. Каждая смена потенциального сигнала в обмотке головки приводит к изменению полярности намагничивания участка дорожки на магнитном носителе, проходящего под головкой.

Считывание информации производится при прохождении под головкой дорожки носителя с записанной информацией. Образующийся проходящим под головкой намагниченным участком дорожки магнитный поток частично замыкается через сердечник головки, пронизывая ее обмотку. При прохождении под головкой границ участков с разной полярностью намагничивания потокосцепление обмотки головки меняется и в ней возникают импульсы той или иной полярности, которые в соответствии с используемым методом записи информации воспринимаются как 1 или 0.

### *Основные характеристики ВЗУ*

Одной из основных характеристик ВЗУ является *общая емкость хранимой информации* или *емкость ВЗУ*, обычно измеряемая в байтах.

Обращение к ВЗУ в общем случае предполагает последовательное выполнение двух процессов:

а) доступа к ВЗУ — подведения головки (головок) к участку носителя, где находится нужная информация или куда информация должна быть записана, или, наоборот, участка носителя к головке (головкам);

б) считывания и передачи информации из ВЗУ в ОП или передачи информации из ОП в ВЗУ и записи ее на носитель.

Соответственно быстрдействие ВЗУ определяется двумя показателями — *средним или максимальным временем доступа* и *скоростью передачи информации (скоростью записи-считывания)*.

На носителе, как правило, информация располагается упорядоченно, поэтому оказывается целесообразным производить запись или считывание не отдельного слова или байта, а последовательно располагаемого на носителе блока или массива данных. Этим достигается уменьшение влияния времени доступа на временные характеристики обмена информацией между ОП и ВЗУ.

Внешние ЗУ делятся на устройства с прямым и последовательным доступом. В *устройствах с прямым доступом*, к которым относятся устройства с магнитными дисками и барабанами,

время доступа (обычно несколько десятков миллисекунд) практически мало зависит от положения носителя относительно головки (головок) в момент инициирования обращения к ВЗУ, что достигается циклическим движением носителя с большой скоростью относительно головки (головок). В устройствах с *последовательным доступом* (ВЗУ на магнитных лентах) для поиска нужного участка носителя требуется последовательный просмотр записанной на носителе информации, для чего в неблагоприятных случаях расположения головок и актуального участка магнитной ленты может потребоваться несколько минут.

К важным характеристикам ВЗУ следует также отнести достоверность функционирования и относительную стоимость устройств.

Обычно *достоверность работы ВЗУ* оценивают числом правильно воспроизводимых в режиме записи-считывания двоичных знаков на один ошибочный знак. Например, современные ВЗУ на магнитной ленте обеспечивают более  $10^9$ , на жестких дисках более  $10^{12}$  правильно записанных и считанных знаков на один ошибочный знак.

*Относительная стоимость ВЗУ* определяется как отношение стоимости устройства к его емкости.

*Плотность записи информации.* Основные характеристики ВЗУ (емкость, скорость передачи данных, относительная стоимость, размеры устройства и др.) прямо зависят от плотности записи информации на носитель, поэтому одним из важнейших направлений улучшения характеристик ВЗУ является повышение плотности записи, что представляет собой сложную инженерную проблему, решение которой связано с улучшением конструкции и технологии изготовления основных узлов ВЗУ, в первую очередь носителя и магнитных головок, создание новых методов магнитной записи и способов кодирования записываемой информации, обеспечивающих корректирование ошибок при считывании.

*Поверхностная плотность записи информации* может быть выражена формулой

$$\delta_s = \delta_l \delta_q,$$

где  $\delta_l$  — *продольная плотность записи*, равная числу бит, записываемых на единицу длины дорожки;  $\delta_q$  — *поперечная плотность записи*, равная числу дорожек, приходящихся на единицу длины в направлении, перпендикулярном движению носителя.

Допустимая продольная плотность записи зависит от характеристик магнитного носителя, зазора между носителем и головкой, конструкции головки, способа записи информации и других факторов. Увеличение поперечной плотности записи можно

достигнуть уменьшением ширины дорожки и уменьшением расстояния между дорожками. Минимальная ширина дорожки ограничивается технологическими трудностями обработки тонкого сердечника для головок. Для повышения поперечной плотности применяют головки, изготовленные на основе тонкопленочной технологии. Следует учитывать, что при уменьшении расстояния между дорожками увеличиваются перекрестные электромагнитные наводки в головках.

Наибольшую плотность удается получить при контактной записи, когда магнитный носитель непосредственно соприкасается с головкой. Такой способ работы применяется главным образом в устройствах с магнитными лентами и гибкими дисками.

При контактной записи трение между магнитным носителем и головкой, вызывая их износ, ограничивает допустимую скорость движения носителя относительно головки. Вместе с тем скорость носителя влияет на такие важные характеристики ВЗУ, как время доступа и скорость передачи информации (скорость записи-считывания). С увеличением этой скорости время доступа к ВЗУ уменьшается, а скорость передачи информации увеличивается.

Применение бесконтактной записи с небольшим зазором между головкой и носителем позволяет создавать ВЗУ с более высокой скоростью передачи информации путем значительного увеличения линейной скорости движения носителя (до нескольких десятков метров в секунду).

Бесконтактную запись широко применяют в устройствах с магнитными дисками и барабанами. Для поддержания плотности на высоком уровне при бесконтактной записи стремятся работать с возможно меньшим зазором. Однако этому препятствуют механические неточности изготовления дисков и барабанов (биение поверхности носителя, эксцентриситет), а также температурные деформации.

Зазор удается значительно уменьшить при использовании *плавающих головок*. В этом случае головку укрепляют в подвижном башмаке, снабженном пружиной, стремящейся прижать его к носителю. Однако в клинообразном воздушном зазоре, образованном башмаком и движущейся поверхностью носителя, возникают аэродинамические силы, препятствующие соприкосновению головок с носителем.

Устанавливается подвижное равновесное состояние, при котором зазор между головкой и поверхностью носителя составляет 2—3 мкм, а в устройствах, разработанных в последние годы, 0,2—1 мкм. Башмак «плывет» около поверхности носителя, следуя за ее биением. При использовании плавающих головок удается работать с продольной плотностью записи 80—

160 бит/мм, а в новейших конструкциях дисковых ЗУ, где зазор составляет 0,2—1 мкм, плотность достигает 400—600 бит/мм.

*Внешние ЗУ на магнитной ленте.* Принцип действия ВЗУ на магнитной ленте поясняет рис. 5.1. В этих устройствах применяют контактный способ записи. Магнитная лента 1 движется только во время подвода к головкам нужного ее участка (зоны), записи и считывания информации, в остальное время лента неподвижна. В ленточных ЗУ большой емкости информация записывается на ленту на нескольких дорожках. Широко используются устройства, работающие с лентой шириной 12,7 мм, с записью информации на 9 дорожках (одна из них — контрольная). В выпускаемых промышленностью устройствах с магнитной лентой продольная плотность обычно составляет 32 и 63, а в новых конструкциях 250—400 бит/мм.

Блок магнитных головок 2 содержит несколько головок, расположенных по одной линии, перпендикулярной направлению движения ленты. Каждой головке соответствует расположенная под ней на ленте дорожка записи. При помощи блока головок одновременно записывается поперек ленты двоичный код (обычно код байта).

В ВЗУ на лентах применяют универсальные, но большей частью сдвоенные головки. В первом случае одни и те же головки используют для считывания и для записи, во втором имеются отдельные блоки головок записи и считывания, расположенные на небольшом расстоянии друг от друга по направлению движения ленты. В этом случае головки считывания используют также для контроля правильности записи информации путем ее считывания сразу после записи.

Лента приводится в движение быстродействующим *старт-стопным лентопротяжным механизмом*. Скорость движения ленты относительно головок должна поддерживаться постоянной, чтобы сохранялись постоянными плотность записи и амплитуда сигналов при считывании информации. Время пуска и останова ленты должно быть минимальным, чтобы можно было работать с короткими промежутками между зонами, в которых записана информация. В противном случае площадь носителя будет использоваться неэффективно. В современных устройствах скорость движения ленты в режимах записи и считывания обычно составляет 2—5 м/с, а время пуска и останова не превышает 5 мс.

Лентопротяжный реверсивный механизм состоит из одного ведущего ролика (вала) 3, приводимого во вращение малоинерционным реверсивным двигателем 4, катушек с лентой 5 с их реверсивными двигателями 6 и вакуумных карманов 7, которые снабжены фотодиодным датчиком 8 положения ленты.

Быстрые пуски, остановки, реверсы, а также протягивание



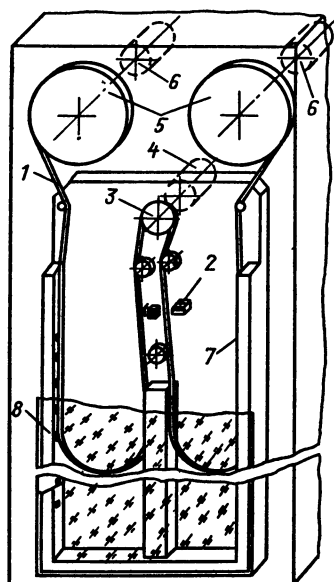


Рис. 5.1. ВЗУ на магнитной ленте (одновальный лентопротяжный механизм)

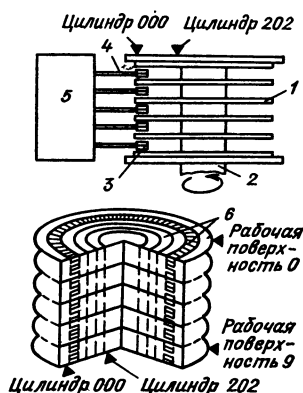


Рис. 5.2. ВЗУ на магнитных дисках:  
1 — диск; 2 — вал; 3 — плавающая головка записи-считывания; 4 — пружинный рычаг; 5 — каретка; 6 — дорожка

ленты по головке обеспечиваются малоинерционным ведущим роликом. Магнитная лента сцепляется с ведущим роликом только за счет трения между его поверхностью и поверхностью ленты. Для увеличения трения угол охвата ролика лентой делается достаточно большим и, кроме того, поверхность ролика покрывают материалом с высоким коэффициентом трения. Двигатель ролика управляется системой, поддерживающей при помощи тахометрических датчиков (на рисунке не показаны) постоянную скорость движения ленты.

Во время движения ленты двигатели 6 вращают катушки так, чтобы с одной катушки лента сматывалась, а на другую наматывалась. При пуске и останове лента движется с большими ускорениями. Катушки с лентой имеют сравнительно большой момент инерции, и привод катушек не может в этих условиях поддерживать натяжение ленты в требуемых пределах. Чтобы избежать недопустимых натяжений и разрывов ленты, а также образования больших петель, нарушающих работу лентопротяжного механизма, между быстродействующим стартстопным механизмом привода ленты и катушками с их инерционным приводом устраивают буферные хранилища в виде *вакуумных карманов*, в которых создается разрежение, удерживающее в карманах слабо натянутую петлю ленты.

*Внешние ЗУ на магнитных дисках.* Принцип построения одного вида такого ЗУ показан на рис. 5.2. Устройство содержит пакет тонких (2—2,5 мм) алюминиевых дисков 1, покрытых тонким слоем магнитного материала. Пакет дисков устанавливается на вал 2 и приводится во вращение электродвигателем. Обе поверхности дисков являются рабочими. Обычно только наружные поверхности крайних дисков не используются для хранения информации.

Информация записывается на концентрических дорожках 6. Дорожки одного и того же диаметра на разных дисках образуют как бы концентричные цилиндры. Понятие *цилиндра* является важным для организации данных на диске.

Чтобы избежать износа носителя и головок, применяют бесконтактную запись и используют плавающие головки. Головки записи-считывания 3 (по одной головке на каждую рабочую поверхность диска) перемещаются в зазорах между дисками. Головки установлены на пружинных рычагах 4, укрепленных в каретке 5. Электрический линейный двигатель с большой точностью преобразует цифровой код номера цилиндра в соответствующее перемещение каретки по радиусу дисков. В результате перемещения головки устанавливаются на нужный цилиндр.

Дисковые ЗУ различного конструктивного исполнения являются наиболее распространенными и совершенными ВЗУ с прямым доступом. По сравнению с магнитным барабаном устройство с дисками в том же физическом объеме имеет во много раз большую поверхность носителя информации.

В настоящее время в вычислительной технике широко применяются следующие типы дисковых ЗУ: а) со сменными пакетами жестких дисков; б) со сменными дисковыми модулями; в) с несменными жесткими дисками; г) со сменными гибкими дисками (дискетами).

*Запоминающие устройства со сменными пакетами жестких дисков* — распространенный тип ЗУ прямого доступа в ЭВМ ЕС средней и большей производительности. Пакеты содержат стандартные диски диаметром 356 мм, изготавливаемые из алюминиевого сплава и покрытые ферролаком. Характерными устройствами этого типа дисковых ЗУ являются устройства емкостью 29 и 100 Мбайт, содержащие соответственно 11 и 12 дисков и работающие с продольной (бит/мм) и поперечной (дорожек/мм) плотностью соответственно 80/4 и 160/8. В первом устройстве установка головок на заданный цилиндр осуществляется разомкнутой системой позиционирования, точность работы которой зависит от точности изготовления узлов, фиксирующих положения блока магнитных головок, и не позволяет работать

с поперечной плотностью свыше 4—5 дорожек/мм. В дисковом устройстве емкостью 100 Мбайт поперечная плотность увеличена до 8 дорожек/мм благодаря применению замкнутой системы позиционирования блоков головок с использованием одной поверхности диска в пакете («сервоповерхности» с нанесенными на нее «серводорожками» и соответственно «сервоголовки» для выработки сигналов отклонения блока головок от заданного положения (заданного цилиндра).

*Запоминающие устройства со сменными дисковыми модулями (типа «Винчестер»).* С увеличением поперечной и продольной плотностей записи усложняется проблема обеспечения совместимости сменных пакетов дисков. Это послужило причиной разработки ЗУ со сменными дисковыми модулями, каждый из которых представляет собой съемный герметизированный контейнер, содержащий пакет дисков вместе с кареткой и головками (в некоторых моделях применены тонкопленочные головки). При такой конструкции продольную плотность записи удастся повысить до 250—600 бит/мм, а поперечную — до 12—25 дорожек/мм и получить сменные модули емкостью 35 Мбайт с 2 дисками (3 информационные поверхности), 75 Мбайт с 4 дисками (6 информационных поверхностей) и 570 Мбайт (11 информационных поверхностей).

В дальнейшем термин «Винчестер» закрепился за ЗУ с несменными жесткими дисками.

*Запоминающие устройства на гибких магнитных дисках* — дешевые и малогабаритные ЗУ, широко используются в качестве внешней памяти персональных компьютеров, малых и микро-

Т а б л и ц а 5.1. Характеристики внешних ЗУ ЕС ЭВМ

Параметр	ЗУ на магнитных лентах		ЗУ на сменных магнитных лентах		
	ЕС-5025	ЕС-5027	ЕС-5061	ЕС-5066	ЕС-5067
Емкость, Мбайт	20 и 40	120	29	100	200
Продольная плотность записи, бит/мм	32 и 63	63 и 246	60—90	160	160
Метод записи информации (см. § 5.3)	БВН-1 и ФК	ФК	ЧМ	ЧМ, МФМ	ЧМ, МФМ
Среднее время доступа, мс	—	—	75	32,5	30
Скорость передачи данных, Кбайт/с	64 и 126	750	312	806	806
Частота вращения пакета дисков, об/мин	—	—	2400	3600	3600

ЭВМ, специализированной памяти для хранения микропрограмм и диагностических тестов, сложных (интеллектуальных) терминалов (графические дисплеи и построители, бесперфокарточные устройства подготовки и ввода данных и т. п.). Особенности конструкции ЗУ с гибкими дисками излагаются в § 5.5.

*Совместимость внешних ЗУ* является одной из важнейших эксплуатационных характеристик. Это свойство позволяет использовать катушки лент, сменные пакеты дисков, дискеты с информацией, записанной на одном ВЗУ и на других аналогичных ВЗУ, работающих в составе различных ЭВМ и систем. Благодаря совместимости ВЗУ имеется возможность обмена информацией, записанной на магнитных носителях.

Для совместимости ВЗУ необходимо выполнение ряда конструктивных и технологических требований. Эти требования сводятся к стандартизации (в ряде случаев в международном масштабе) размеров катушек, пакетов дисков, дискет, материала и технологии изготовления носителей информации, размеров и характеристик магнитных головок, лентопротяжных механизмов, способов и плотности записи, размещения информации на носителе, способов контроля информации.

В табл. 5.1 приведены основные характеристики некоторых электромеханических ЗУ Единой системы ЭВМ.

### **5.3. Методы записи информации на магнитный носитель**

В электромеханических ЗУ используется ряд методов записи информации на носитель. Во всех методах запись производится потенциальными сигналами, поступающими в головку записи. При выборе метода записи учитываются следующие факторы: возможность получения высокой плотности записи, помехоустойчивость метода, обладает метод свойством самосинхронизации при считывании или требует внешней синхронизации, необходимо предварительное стирание ранее записанной информации или допустима запись новой информации без предварительного придания участку носителя исходного магнитного состояния, сложность электронных схем обработки сигналов, считываемых с носителя.

Удешевление электронных схем делает возможным применение более эффективных, хотя и более сложных методов кодирования записываемой информации и процедур обработки считываемых сигналов, в том числе процедур контроля и коррекции ошибок при считывании информации.

*Метод записи без возврата к нулю с переключением потока по единицам (БВН-1).* По этому методу запись производится без возврата носи-

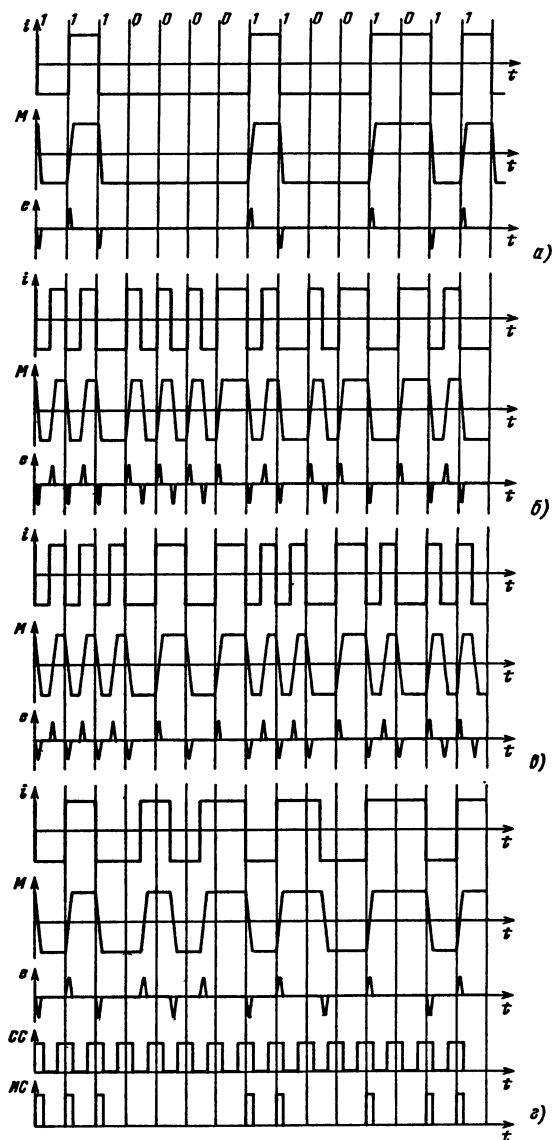


Рис. 5.3. Методы магнитной записи цифровой информации:  
*а* — запись без возврата к нулю с переключением потока по единицам (БВН-1);  
*б* — запись с фазовым кодированием; *в* — запись с частотной модуляцией;  
*г* — запись с модифицированной фазовой модуляцией; *i* — ток в головке записи;  
*M* — намагниченность магнитного носителя; *e* — ЭДС считывания; *СС* — синхросигнал; *ИС* — сформированный информационный сигнал

теля к исходному состоянию (размагниченному состоянию или состоянию насыщения определенного знака).

Диаграмма для метода записи БВН-1 приведена на рис. 5.3, а. При записи 1 ток в обмотке записи изменяет направление, и носитель соответственно переходит из состояния насыщения одного знака в состояние насыщения другого знака. При записи 0 направление тока в обмотке и состояние носителя не меняются. Достоинством метода является возможность записи на носитель новой информации без стирания предыдущей.

При считывании 1 в обмотке головки индуцируются импульсы (разнополярные), а при считывании 0 сигнал с дорожки не поступает. Поэтому для распознавания информации при считывании необходимы синхросигналы, задающие границы каждого такта записи. Признаком 1 является совпадение информационного импульса, считанного с дорожки на ленте, с синхросигналом, а признаком 0 — наличие только синхросигнала (отсутствие информационного импульса).

Стандартом предусмотрено применение метода БВН-1 при записи информации на магнитных лентах с плотностью 8 и 32 бит/мм. На лентах производится многодорожечная запись, и в каждой позиции (строке) записывается поперек ленты байт с дополнительным разрядом контроля по нечетности (см. гл. 12). Поэтому при считывании любой строки хотя бы с одной дорожки поступает сигнал 1, чем достигается самосинхронизация считываемой информации.

*Метод записи с фазовым кодированием (ФК) (фазовой модуляцией).* Диаграмма для этого метода приведена на рис. 5.3, б. На границе каждого такта записи происходит смена направления тока в записывающей головке и, следовательно, смена магнитных состояний носителя. Полярность тока изменяется в одном направлении при записи 0 (например, от отрицательной полярности к положительной) и в противоположном направлении при записи 1. Происходит как бы изменение фазы тока записи. Логическая схема тракта записи анализирует значение следующего записываемого двоичного знака. Если должен быть записан тот же знак, что и в предыдущем такте, то в середине такта изменяется направление тока записи в головке. Если должен быть записан другой знак, изменение направления тока посередине такта не производится. Так как при записи последовательности одинаковых знаков в серединах тактов производится дополнительное переключение направления тока, то частота изменения тока записи в этом случае увеличивается в 2 раза по сравнению с частотой изменения тока при записи последовательности неодинаковых знаков.

При считывании 1 и 0 распознаются по полярности импульса ЭДС в головке считывания в первом полутакте.

Несмотря на увеличение частоты переключения тока записи, метод ФК по сравнению с методом БВН-1 позволяет работать с большей плотностью записи, так как на фазовые соотношения сигналов оказывают малое влияние помехи, возникающие из-за наложения полей, создаваемых соседними участками носителя. Метод ФК является самосинхронизирующимся (в первом полутакте всегда имеется импульс той или иной полярности), не требующим предварительного стирания ранее записанной информации.

В соответствии со стандартом метод ФК применяется при записи информации на магнитные ленты с плотностью 63 бит/мм. В некоторых ЭВМ предусматривается возможность работы устройств с магнитной лентой при плотностях записи информации 32 и 63 бит/мм и соответственно с использованием методов БВН-1 и ФК, причем переключение

с одного режима при записи (или считывании) на другой производится автоматически программным путем.

*Метод записи с частотной модуляцией (ЧМ).* Метод поясняется на рис. 5.3, в. Ток записи изменяет направление на границе каждого такта записи и, кроме того, посередине такта при записи 1. Запись производится под управлением вырабатываемой в ЗУ серии синхросигналов, определяющих границы тактов записи. При записи 1 в середине такта поступает дополнительный импульс, вызывающий дополнительное переключение тока в головке.

Таким образом, при записи 1 частота переключения тока записи вдвое больше, чем при записи 0. Поэтому этот метод называют также двухчастотным. Считыванию 1 соответствует наличие импульса произвольной полярности во втором полутакте, а считыванию 0 — его отсутствие.

Метод ЧМ не требует перед записью предварительного стирания информации и является самосинхронизирующимся (при считывании в первом полутакте всегда присутствует импульс той или иной полярности), что особенно важно для ЗУ на дисках, где применяется односторонняя запись.

Применение метода записи ЧМ предусмотрено международным стандартом для ЗУ со сменными пакетами дисков.

*Метод записи с модифицированной фазовой модуляцией (МФМ) (трехчастотный)* (рис. 5.3, г) получил в последнее время распространение главным образом в ЗУ со сменными дисками большей емкости (100—300 Мбайт). Метод обеспечивает самосинхронизацию сигналов считывания и позволяет получить большую плотность записи, чем другие методы. Записью управляет серия синхросигналов, задающих ее такты. Переключение тока в головке происходит при записи 1 в начале записи, а при записи 0 — посередине такта, если только следующий записываемый знак не 1. В противном случае при записи 0 переключение тока блокируется.

При записи последовательности одинаковых двоичных знаков 0 или 1 частота переключений тока записи совпадает с частотой синхросигналов. При записи комбинации 10 или 01 интервал между переключениями тока увеличивается до полутора периодов синхросигналов, а при записи 101 — до двух. Таким образом, в последовательности переключений тока записи и соответственно в последовательности смены полярности намагничивания участков дорожки носителя имеются три частоты. При считывании каждое изменение полярности намагничивания участков дорожки индуцирует в обмотке головки импульс той или иной полярности. Этот импульс соответствует считанной 1, если он совпадает по времени с синхросигналом, и 0 если не совпадает.

## **5.4. Представление информации на магнитных лентах и дисках (ЕС ЭВМ)**

*Представление информации на магнитных лентах.* В ВЗУ на магнитных лентах применяется главным образом последовательно-параллельное размещение информации. Строка информации, записанная поперек ленты, соответствует одному машинному слогу (байту). Машинное слово занимает на ленте несколько строк (рис. 5.4).

*Блоки информации,* т. е. данные, располагаемые на носителе компактно (без промежутков) и записываемые и считываемые одной командой ЭВМ, помещаются в отдельные зоны. Зоны разделяются между собой

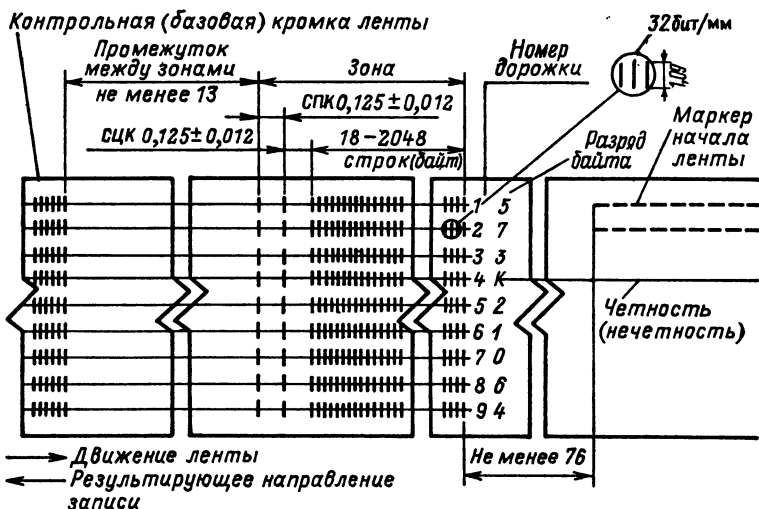


Рис. 5.4. Размещение информации на магнитной ленте при плотности записи 32 бит/мм (размеры указаны в миллиметрах): СЦК — строка циклического контроля; СПК — строка продольного контроля

промежутками. Межзональные расстояния выбираются таким образом, чтобы пуск и останов ленты не вызывали потери информации.

При записи на магнитную ленту в ряде случаев используют зоны (блоки) фиксированной длины. Если в зоны фиксированной длины приходится записывать разное количество информации, то использование поверхности носителя будет неполным.

Более эффективными являются зоны переменной длины, размеры которых устанавливаются программой в процессе решения задачи. Следует отметить, что запись очень короткими зонами приводит к плохому использованию поверхности магнитного носителя, так как значительную часть площади носителя будут занимать промежутки между зонами.

На ленту помимо данных записывается также служебная информация: признаки начала и конца зоны, коды для контроля правильности записи и считывания информации (см. гл. 12).

Применяется контроль правильности считанной информации по строке (поперечный контроль) и правильности в зоне (продольный контроль). Поперечный контроль обычно осуществляется по нечетности суммы всех 1 в разрядах строки, что дает возможность обнаружить в строке при считывании любую одиночную ошибку. Для контрольного разряда нечетности выделяется специальная дорожка. При продольном контроле по четности в конце каждой зоны записывается отдельная контрольная строка — строка продольного контроля (СПК) — разряды которой дополняют сумму единиц по соответствующим дорожкам по четности.

В ряде случаев, например при записи информации на магнитные ленты, предназначенные для обмена между различными потребителями, применяется дополнительный, так называемый циклический контроль. Строка циклического контроля (СЦК) размещается в конце зоны перед



СПК. Циклический контроль дает возможность обнаруживать и исправлять при считывании одиночные и групповые ошибки по одной дорожке (см. [27]). Строка циклического контроля и СПК формируются в процессе записи устройством управления ленточного ЗУ.

Чтобы различать начало или конец зоны, применяют различные способы. Одним из способов является использование специальных кодовых комбинаций, записанных на ленте. В устройствах, использующих поперечный контроль по нечетности, в каждой строке записана по крайней мере одна 1. Граница между зонами в таких устройствах устанавливается по отсутствию каких-либо сигналов с носителями в течение определенного времени.

При подходе магнитной головки к физическому краю ленты (началу или концу) необходимо выработать сигналы для автоматического останова ленты. С этой целью на ленту наносят тем или иным способом маркеры начала и конца ленты<sup>1</sup>.

Для обеспечения совместимости ВЗУ на магнитных лентах разработан международный стандарт формата размещения информации на них (рис. 5.4). Стандарт предписывает использование ленты шириной 12,7 мм с девятью дорожками для записи информации и контрольного разряда поперечной нечетности. Толщина ленты 48 мкм, длина ее на катушке около 750 м. Стандарт предусматривает запись с плотностью 8 и 32 бит/мм по методу БВН-1 или 63 бит/мм по методу ФК. На рис. 5.4 показан формат размещения информации на ленте при плотности записи 32 бит/мм. Строка циклического контроля записывается в конце зоны после последнего байта данных с промежутком четыре строки. Затем с тем же промежутком записывается СПК. При плотности 8 и 63 бит/мм формат размещения данных на ленте аналогичен показанному на рис. 5.4, но при этом СЦК отсутствует.

В современных ЭВМ информация на носителях, в том числе и на магнитных лентах, хранится, как правило, в виде *файлов*. Файлом называется совокупность записей, объединенных по некоторому общему смысловому признаку.

Запись, входящая в файл, может занимать целиком блок информации на носителе либо несколько записей могут быть объединены в одном блоке. В последнем случае говорят о так называемой *блокированной записи*.

При организации хранения и обработки данных, располагаемых в ВЗУ, используется понятие тома. *Том* — это стандартный для данного устройства типоразмер носителя информации (катушка магнитной ленты, пакет дисков, дискета и т. п.).

На носитель помимо основной информации записывают служебную — так называемые метки. Метки представляют собой специального вида записи, описывающие характеристики томов и файлов. Кроме того, на ленту заносятся специальные записи для отделения друг от друга различных видов информации — разделительные маркеры.

На рис. 5.5 показано в упрощенной форме представление на ленте файлов в виде многофайлового тома.

В начале тома располагается метка *Начало тома*, описывающая данную катушку ленты (номер тома в данном вычислительном центре, имя владельца и т. п.).

---

<sup>1</sup> Обычно маркерами служат наклеенные в начале и конце ленты полоски фольги, отражающие свет на маркерный фотодиод.

Далее следует метка *Начало файла*, которая указывает наименование файла, номер тома, в котором размещается файл, номер модификации файла в процессе его обработки или подновления, дата создания файла и срок его хранения, секретность информации и др. Окончание файла отмечается меткой *Конец файла*, которая подобна начальной метке, но, кроме того, обычно указывает количество записей с основной информацией в данном файле.

Служебные записи (метки) отделяются от основных записей разделительными маркерами. После окончания файла на катушке обычно записываются подряд два маркера.

В ВЗУ на магнитных лентах реализуются операции считывания (при движении ленты в прямом и обратном направлениях), записи, стирания блоков информации, передвижения (вперед и назад) на один блок (одну зону) или до следующего разделительного маркера, перемотки ленты в исходное состояние. Используя эти операции, а также данные, хранимые в служебных метках, программа может осуществлять поиск и обработку записей внутри файла и поиск и идентификацию томов и файлов.

*Представление информации на магнитных дисках.* Запись информации на диск производится обычно последовательным кодом на concentрические дорожки на поверхности диска. Группа байтов, записанная последовательно вдоль дорожки, образует блок информации, отделяемый от следующего блока на этой дорожке промежуточком. В современных ВЗУ на магнитных дисках используются блоки как фиксированной длины — секторы (для относительно простых и дешевых устройств), так и переменной.

Расположение файла на одном цилиндре позволяет производить поиск и обработку записей, входящих в этот файл, без радиального перемещения головок, что существенно сокращает время доступа к данным.

В качестве примера организации данных на диске рассмотрим типичное расположение информации на дорожке для устройства с блоками переменной длины (рис. 5.6). Начало дорожки распознается с помощью метки начала оборота, которая выполняется обычно в виде прорези на специальном секторном диске.

В начале каждой дорожки помещается служебный блок, называемый *Собственным адресом*. Собственный адрес определяет номер цилиндра и номер магнитной головки внутри цилиндра для данной дорожки.

Вслед за собственным адресом размещается служебный блок *описание дорожки*, описывающий степень заполнения дорожки данными, адрес

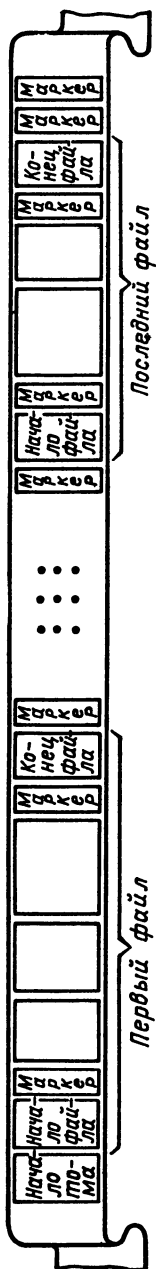


Рис. 5.5. Структура файлов на магнитных лентах

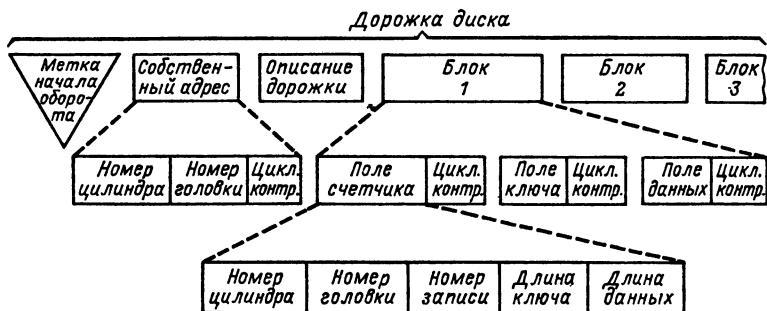


Рис. 5.6. Представление информации на магнитных дисках

запасной дорожки, если данная дорожка повреждена, и др.

Далее располагаются блоки, содержащие рабочие записи. Эти блоки состоят из трех полей: поля счетчика, поля ключа и поля данных.

Поле счетчика содержит служебную информацию, необходимую для обработки записей, хранимых в блоке. В этом поле указываются номера цилиндра и головки, которые в случае исправности дорожки совпадают с соответствующими номерами в поле собственного адреса. Если дорожка неисправна, то указывается номер соответствующей альтернативной (запасной) дорожки. На альтернативной дорожке записываются номера цилиндра и головки основной дорожки.

Поле счетчика указывает также номер записи и длину остальных двух полей — поля ключа и поля данных.

Поле ключа предназначается для информации, идентифицирующей данную запись (например, имя файла или какие-либо другие смысловые признаки).

Поле данных хранит информацию, непосредственно образующую саму запись или группу записей. Поле ключа или поле данных может отсутствовать, при этом в поле счетчика указывается, что длина соответствующего поля равна нулю.

В конце служебных полей и полей данных записываются контрольные циклические коды, позволяющие при считывании проверять правильность записанной информации.

Рассмотренная выше структура информации на диске позволяет строить файлы различных видов, при этом часть рабочих записей используется для представления меток тома, файла, маркеров и других идентификаторов, а в остальные записи заносятся запаасаемые на диске данные.

В современных ЭВМ предусматривают широкий набор операций (приказов) в ВЗУ на магнитных дисках: установку цилиндра, головки; поиск информации по собственному адресу, по номеру цилиндра и головки, номеру записи, ключу, данным; чтение, запись различных полей на диске (собственного адреса, ключа, поля данных и др.).

## 5.5. Периферийные устройства персональных компьютеров

### 5.5.1. ЗУ на гибких и жестких магнитных дисках персональных компьютеров

Периферийные устройства персональных компьютеров представляют собой многообразную специфическую ветвь в технике периферийных устройств ЭВМ, для которой характерным является стремление к уменьшению габаритных размеров устройств, высокой надежности, технологичности и приспособленности к массовому производству, сравнительной дешевизне, а главное, к созданию удобств и обеспечению высокой эффективности при взаимодействии пользователя с ПК [20, 21, 44].

Запоминающие устройства на гибких магнитных дисках (ЗУГД). В составе персональных компьютеров ЗУГД служит внешней сменной памятью, используемой для длительного хранения программ и результатов обработки данных на ПК, а также для обмена информацией между пользователями.

Носителем информации является лавсановый покрытый ферролаком круглый диск (дискета), постоянно заключенный в квадратную пластмассовую картонную кассету, имеющую внутри прокладки для уменьшения трения дискеты о стенки кассеты (рис. 5.7). Информация записывается на нескольких концентрически расположенных круговых дорожках. Дискета

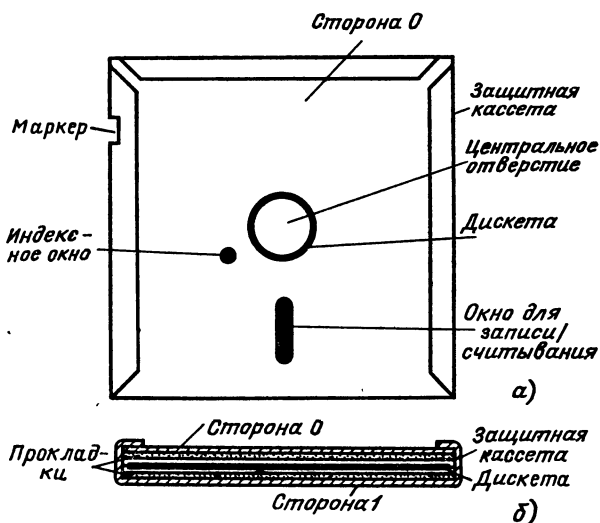


Рис. 5.7. Кассета с гибким магнитным диском (дискетой):  
а — вид со стороны 0; б — вид в разрезе

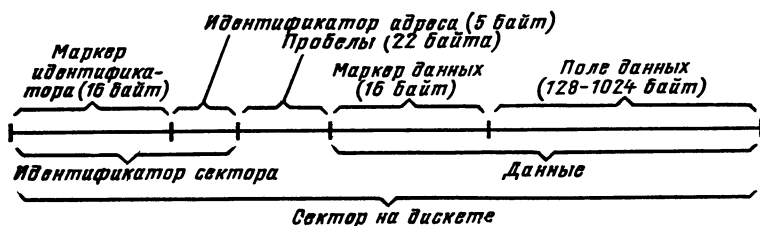


Рис. 5.8. Структура сектора на дискете

снабжена центральным отверстием для установки на ступице дисководов. Используется контактная запись. Для уменьшения износа головки из-за трения поверхность дискеты покрывается специальным лаком.

В кассете имеется окно, через которое магнитная головка соприкасается с поверхностью вращающейся дискеты, а также «индексное отверстие», определяющее начало дорожек.

Размеры дискеты стандартизированы. Ее толщина 0,12 мм, внешний диаметр в зависимости от типа дискеты составляет 133 или 89 мм (соответственно 5,25 или 3,5 дюйма).

Кассета снабжена выемкой — маркером разрешения записи. Если выемка заклеена — запись запрещена.

Употребляются дискеты с одинарной или двойной (поперечной) плотностью записи, односторонние работают с одной головкой, двусторонние (используются обе поверхности дискеты) — с двумя головками. В зависимости от этих факторов, а также формата записи полезная емкость ЗУГД составляет 160 Кбайт — 2 Мбайт.

В ПК обычно используются два формата записи на дискеты — с 40 и 80 дорожками. Дорожки содержат 8 или 9 секторов, а в секторе может находиться 128, 256, 512 или 1024 байт данных (рис. 5.8). Число дорожек, секторов, байт данных в секторе устанавливается программно при «форматировании» дискеты, так же как и скорость перехода головки с дорожки на дорожку.

Запоминающее устройство на гибком магнитном диске представляет собой довольно простой механизм (рис. 5.9), так как из-за низкой плотности записи (сравнительно с ЗУ с жесткими дисками) не предъявляются высокие требования к скорости и точности установки головки на дорожку. Дискета 3 в кассете 5 устанавливается на ступицу 7 и крепится пружинным прижимом 4. Ступица приводится во вращение электродвигателем при помощи ременной передачи 6. Каретка 8 с магнитной го-

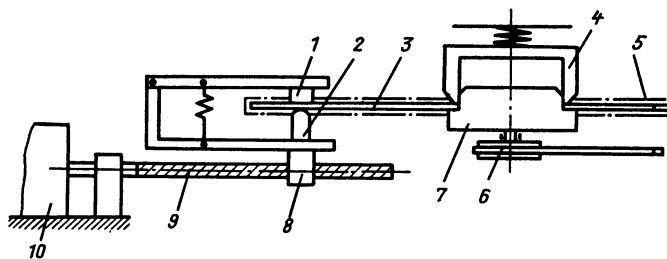


Рис. 5.9. Механизм ЗУ на гибких магнитных дисках (ЗУГД)

ловкой 2 и пружинным фиксатором 1 перемещается шаговым двигателем 10 посредством винтовой передачи 9.

Частота вращения дискеты обычно 90—360 об/мин, среднее время доступа 0,1 с. Продольная плотность записи 10—128 бит/мм, скорость передачи данных около 0,25 Мбит/с. Из-за износа носителя и головок регламентируется общее число проходов носителя над головкой, которое для современных конструкций составляет обычно до  $5 \cdot 10^6$ .

*Запоминающие устройства на жестких несменных дисках (ЗУЖД).* В составе ПК ЗУЖД служит внешней несменной памятью для длительного хранения больших емкостей информации (программ и данных, включая полученные на ПК результаты вычислений).

В ПК используются ЗУЖД с жесткими дисками (типа «Винчестер»), главным образом диаметром 133, а также 89 мм, со стандартной и половинной высотами устройства, работающие с высокой плотностью записи и имеющие большую емкость информации (от 5 до 120 Мбайт), небольшое среднее время доступа (0,025 с) и высокую скорость передачи данных (1—8 Мбайт/с).

*Электронные квазидиски.* В целях повышения надежности работы внешней памяти, в первую очередь, путем исключения движущихся механизмов, а также для уменьшения габаритных размеров и повышения быстродействия создаются ЗУ большой емкости на основе БИС памяти или памяти на магнитных доменах со структурой организации информации и процедурами доступа, моделирующими структуры и процедуры ЗУ на магнитных дисках и выполняющими все их команды (приказы). Такие устройства называют электронными дисками. В ПК часть ОП может использоваться в режиме моделирования работы (но с большим быстродействием) ЗУГД.

*Оптические диски.* В последние годы в качестве постоянных ЗУ очень большой емкости начинают в составе ПК применять

«оптические компакт-диски» (ЗУ на оптических дисках — ЗУОД)<sup>1</sup>.

На специальной производственной установке информация записывается лазерным лучом, выжигающим микрометки 0 и 1 по концентрическим окружностям на стеклянном диске, по которому, как по матрице, изготавливают пластиковые копии, приобретаемые пользователями. Считывание производится лучом малоомощного лазера, который, различным образом (из-за разной отражательной способности) отражаясь от микрометок 0 и 1, воспринимается фотодиодами.

Типичными параметрами ЗУОД являются диаметр диска 120 мм, его толщина 1,2 мм, информационная емкость 550 Мбайт (достаточна для записи полного текста Большой Советской Энциклопедии), скорость передачи данных 1,5 Мбит/с, среднее время доступа (довольно большое) 0,5 с, что объясняется сравнительно большой массой оптической головки считывания.

В последнее время появились ЗУОД, позволяющие пользователю производить лазерным лучом одnorазовую запись и многократное считывание (при сниженной мощности лазера).

Запоминающим устройствам на оптических дисках стараются придать такие размеры, чтобы их можно было устанавливать в ПК вместо ЗУ на магнитных дисках.

### **5.5.2. Клавиатура**

Клавиатура (выносная) служит пользователю основным средством для ввода информации в ПК и ручного задания различных режимов работы и процедур. В аппаратуру, обслуживающую клавиатуру, входит работающий по собственной программе 8-разрядный микропроцессор (точнее, микроконтроллер), например K1816BE48, придающий клавиатуре большую гибкость и программную настраиваемость на выполнение достаточно сложных процедур («интеллектуальная клавиатура»).

На рис. 5.10 представлена клавиатура ПК ЕС-1840, в которой можно выделить три группы клавиш:

а) клавиши для ввода алфавитно-цифровой информации (при работе с русским и латинским алфавитами), а также управляющие клавиши, образующие центральную группу клавиш;

б) клавиши для ввода числовой (цифровой) информации и управления курсором дисплея — правая группа клавиш;

в) левая группа (на рисунке не показана) функциональных клавиш ( $\Phi 1$ — $\Phi 10$ ), которые можно запрог-

---

<sup>1</sup> В зарубежной литературе эти устройства называют Compact Disk Read-Only (CD-ROM).

Для ввода некоторой последовательности прописных букв следует нажать и отпустить клавишу **ФБП** (фиксация букв прописных) и снова ее нажать и отпустить при

КЛЮЧ	!	@	#	\$	%	^	&	*	(	)	_	+	-	.	↵	⊗	ЦИФ	ФСА СТОП
←	Й	Ц	У	К	Е	Т	Г	Ш	Щ	З	Х	Ъ	}	?	ПЕЧ	7	8	9
→	Я	Ч	Ф	В	А	П	Р	О	Л	Д	Ж	Э	{	~	ВВОД	4	5	6
	Ф	А	Ы	Б	И	Г	Н	Д	К	Л	Ж	Э	[	^	ФБН	1	2	3
ДОП	↑	Я	Ч	Х	С	М	В	Б	Н	Ю	~	<	.	↑	⊗	0	КОН	→
⊗	⊗	ЛАТ	Р/Л	ИНФ								Р/Л	РУС	⊗		—	ВСТ	УДЛ

Рис. 5.10. Клавиатура ПК ЕС-1840



	00	10	20	30	40	50	60	70
0		┐		0	Q	P	'	P
1	⊞	←	!	1	A	Q	a	q
2	⊞	⊞	"	2	B	R	b	r
3	♥	!!	#	3	C	S	c	s
4	♣	¶	\$	4	D	T	d	t
5	♣	⊞	%	5	E	U	e	u
6	♣	—	&	6	F	V	f	v
7	•	⊞	'	7	G	W	g	w
8	⊞	↑	<	8	H	X	h	x
9	⊞	↓	>	9	I	Y	i	y
A	⊞	→	⊞	:	J	Z	j	z
B	♂	←	+	;	K	[	k	⊞
C	♀	└	,	<	L	\	l	!
D	┐	⊞	—	=	M	]	m	⊞
E	┐	⊞	.	>	N	^	n	~
F	⊞	♥	/	?	O	_	o	Δ

Рис. 5.11. Кодовая таблица набора знаков ПК ЕС-1840

переходе к вводу строчных букв. В нажатом состоянии клавиш *P/Л* происходит переход к альтернативному алфавиту.

Для управления вводом с клавиатуры и редактирования информации используются управляющие клавиши.

Нажатие клавиши *ВВОД* инициирует ввод в ПК набранного на клавиатуре сообщения. Возврат на символ или несколько символов с их стиранием выполняется соответственно однократным или многократным нажатием клавиши *←*. Клавиша *ПЕЧ* (при работе с верхним регистром) инициирует вывод на печатающее устройство информации с экрана дисплея. Одновременное нажатие клавиш *УПР* и *P* (режим) позволяет предписать посимвольный вывод на экран и печать. Одновременное нажатие клавиш *УПР* и *СТОП* останавливает вывод информации на печатающее устройство.

	80	90	A0	B0	C0	D0	E0	F0
0	≡	Г	Г	А	Р	а	р	Е
1	≡	Г	Г	Б	С	б	с	ё
2	≡	Г	Г	В	Т	в	т	/
3	≡	Г	Г	Г	У	г	у	\
4	≡	Г	Г	Д	Ф	д	ф	/
5	≡	Г	Г	Е	Х	е	х	\
6	≡	Г	Г	Ж	Ц	ж	ц	→
7	≡	Г	Г	З	Ч	з	ч	←
8	≡	Г	Г	И	Ш	и	ш	↓
9	≡	Г	Г	Й	Щ	й	щ	↑
A	≡	Г	Г	К	Ъ	к	ъ	÷
B	≡	Г	Г	Л	Ы	л	ы	±
C	≡	Г	Г	М	Ь	м	ь	№
D	≡	Г	Г	Н	Э	н	э	Ж
E	≡	Г	Г	О	Ю	о	ю	■
F	≡	Г	Г	П	Я	п	я	

и EC-1841

Функциональные клавиши  $\Phi 1$ — $\Phi 10$  позволяют инициировать выполнение закрепленных за ними команд операционной системы, причем пользователь может менять перечень таких команд, может приписать функциональной клавише процедуру ввода фиксированного текста (до 20 символов).

Одновременное нажатие клавиши ДОП и других клавиш позволяет модифицировать выполнение соответствующих операций.

На рис. 5.11 представлены коды набора знаков ПК EC-1840 и EC-1841. Левая часть таблицы соответствует основной таблице кодов ASCII, а правая является ее расширением для основного варианта знакогенератора с русским алфавитом, рекомендуемым для отечественных ПК. При использовании зарубежных ПК предпочтительнее другой, так называемый аль-

тернативный вариант расширения таблицы ASCII для знакогенератора с русским алфавитом [11].

### 5.5.3. Дисплеи

Дисплеем или монитором называется устройство визуального отображения информации на экране. В настоящее время дисплей является наиболее эффективным средством вывода информации при диалоговом взаимодействии пользователя с ЭВМ. Пользователь с помощью клавиатуры (и других средств) может сформировать на экране вводимую информацию, проверить ее, при необходимости отредактировать и затем ввести в машину. Пользователь также может вызвать на экран любую информацию, хранящуюся в памяти машины, при необходимости откорректировать ее и снова отправить в машину для дальнейшей обработки.

К дисплеям, работающим в составе ПК, помимо требования четкого представления текста и графических изображений предъявляются также требования небольших габаритных размеров, небольшой потребляемой мощности и, конечно, небольшой стоимости. Желательным является возможность многоцветного представления информации (в первую очередь изображений), что повышает их наглядность.

Наибольшее распространение получили дисплеи, в которых используются электронно-лучевые трубки (ЭЛТ), принцип действия которых пояснен на рис. 5.12. Экран 1 трубки покрыт с внутренней стороны слоем люминофора. Электронный луч (пучок электронов) 2, попадая в некоторую точку экрана и возбуждая в ней частицы люминофора, формирует световое пятно. Электронный луч создает электронная пушка 3, содержащая подогреваемый катод, модулятор, ускоряющий электрод и фокусирующую систему. Луч направляется в определенную точку экрана магнитным или электростатическим полем, создаваемым отклоняющей системой 4 под воздействием подводимых к ней управляющих напряжений.

По виду представляемой информации различают дисплеи алфавитно-цифровые, квазиграфические и графические. Дисплеи бывают монохромные (черно-белые) и цветные.

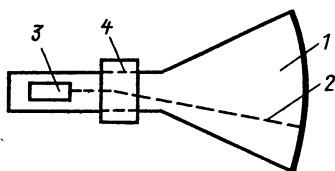


Рис. 5.12. Электронно-лучевая трубка дисплея

*Алфавитно-цифровые дисплеи* предназначены для отображения текстовой информации, при этом используется фиксированный набор символов (буквы, цифры и другие знаки).

В алфавитно-цифровых дисплеях рабочая часть экрана ЭЛТ разбивается на строки, строки разбиваются на *знаковые места*. Знак изображается в пределах предусмотренного знакового места. Знаковое место имеет свою невидимую сетку, по узлам и линиям которой формируется изображение экрана.

В настоящее время наиболее распространенными являются функциональный и растровый способы формирования знаков на экранах дисплеев.

При *функциональном способе формирования знаков* перемещение луча осуществляется двумя электромагнитными отклоняющими системами. Одна из них — знаковая система (ЗС) — перемещает пятно в пределах одного знакового места, другая — координатная система (КС) — после завершения изображения некоторого знака перемещает погашенное пятно в исходную точку соседнего знакового места.

При этом способе управление дисплеем должно обеспечивать перемещение пятна луча с постоянной скоростью по контуру изображаемого знака (рис. 5.13) и его модуляцию в нужные моменты времени. Здесь столбцы знакового места I—IV используются для изображения знака, а V — для формирования промежутка между соседними знаками. На рис. 5.13 цифрами 0—9 отмечена траектория луча при изображении буквы Б.

При *растровом способе формирования знаков* перемещение луча по экрану не зависит от изображаемых знаков и осуществляется всегда по одной и той же сетке. Растровый дисплей выполняется по той же структурной схеме, что и функциональный, но только с одной отклоняющей КС.

При растровом способе на изображение строки текста затрачивается, например, 16 телевизионных линий, из них 7 отводятся для собственно знаков (рис. 5.14), а следующие 9 — для промежутка между строками текста. Пятно луча движется слева направо по всей рабочей части экрана по линии начала Л0, затем Л1 и т. д. Схема синхронизации блока управления дисплеем вырабатывает сигналы линий Л0, Л1, ..., Л6 и сигналы

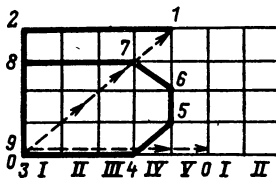


Рис. 5.13. Функциональный способ формирования знаков

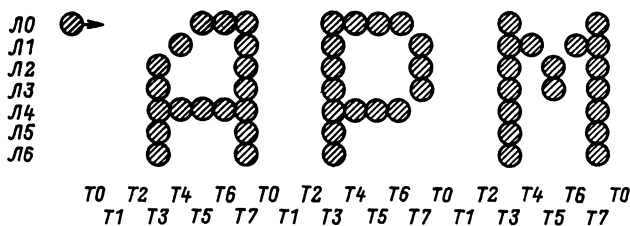


Рис. 5.14. Растровый способ формирования знаков

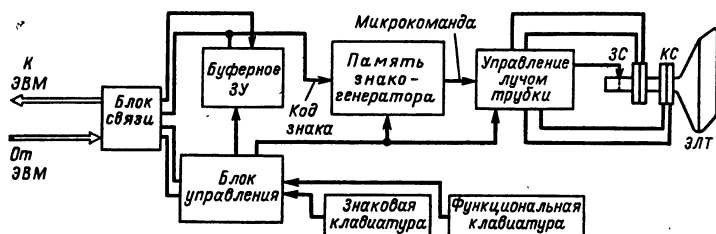


Рис. 5.15. Структурная схема алфавитно-цифрового дисплея (с функциональным способом формирования знаков)

такты  $T_0, T_1, \dots, T_7$  формирования элементов знаков. Такты  $T_0-T_2$  формируют промежуток между знаками,  $T_3-T_7$  участвуют в изображении знака. При движении пятна луча по первой линии ( $L_0$ ) модуляцией луча формируются только верхние элементы всех знаков данной текстовой строки, при движении по второй линии ( $L_1$ ) — вторые элементы и т. д.

Таким образом, для формирования на линии элементов знаков нужно иметь сведения, на какой линии находится пятно луча (сигналы,  $L_0, L_1, \dots, L_7$ ), какой знак изображается (код знака) и на каком столбце знакового места находится пятно (сигналы  $T_3, T_4, \dots, T_7$ ).

Информация (микропрограммы), управляющая модуляцией луча, для всех знаков хранится в постоянном ЗУ<sup>1</sup>. В процессе изображения текста ПЗУ выдает только сигналы-модуляции, соответствующие знаку и положению луча на сетке раstra.

Структурная схема дисплея с функциональным способом формирования знаков приведена на рис. 5.15. Буферное ЗУ (па-

<sup>1</sup> Понятия микропрограммы и микрокоманды поясняются в гл. 6.

мать регенерации) дисплея предназначено для хранения кодов знаков. Емкость ЗУ равна максимальному числу выводимых на экран знаков. Каждому знаковому месту на экране соответствует ячейка в буферном ЗУ. Память *знакогенератора* (ПЗГ) хранит микропрограммы управления изображением на экране для всех знаков используемого алфавита.

Блок управления лучом ЭЛТ под воздействием микрокоманд, поступающих из ПЗГ, вырабатывает аналоговые сигналы для знаковой (ЗС) и координатной (КС) отклоняющих систем, обеспечивающих перемещение, и для модуляции луча, при которых на экране высвечивается рисунок заданного знака, после чего луч устанавливается в исходную точку соседнего знакового места.

При выводе информации на экран по инициативе ЭВМ текст, сформированный в виде последовательности кодов знаков, передается из ЭВМ в буферное ЗУ. Коды знаков в ЗУ располагаются в той последовательности, в какой знаки должны размещаться на экране.

После записи кодов в буферное ЗУ в блок управления поступает приказ *Начать отображение*. Дисплей переходит в автономный режим работы и начинает отображение текста, при этом в порядке размещения кодов в ЗУ производится отработка кодов знаков. Под воздействием сигналов блока управления код очередного знака поступает в ПЗГ микропрограмм знакогенератора.

Знакогенератор воспринимает код как команду, в ответ на которую выдает последовательность микрокоманд управления изображения заданного знака.

Для получения немерцающего изображения во всех дисплеях с ЭЛТ обеспечивается воспроизведение (регенерация) на экране хранящейся в памяти информации (изображения) 50 раз/с.

*Квазиграфические дисплеи* отличаются от алфавитно-цифровых наличием дополнительного набора знаков — укрупненных типовых графических элементов — отрезков прямой линии и дуг, двумерных фигур. Такие графические знаки называют *графемами*. Размер графического знакоместа больше знакоместа обычных текстовых знаков, чтобы имелась возможность формировать из графических элементов цельное графическое изображение. Способ кодирования и отображения на экране графем подобны обычным знакам.

Приведем характерные параметры монохромного (не цветного) алфавитно-цифрового квазиграфического дисплея персонального компьютера ЕС-1840: диагональ экрана 31 см; разрешающая способность 25 строк по 80 знаков в строке; общее число отображаемых знаков (включая графемы) до 256; имеется

возможность путем программного изменения содержимого памяти знакогенератора создавать нужный пользователю набор знаков; емкость памяти регенерации 2 Кбайта.

Устройство управления (адаптер) дисплея содержит не показанную на рис. 5.15 память атрибутов знаков — кодов, задающих способ отображения для каждого знака, в том числе графем (нормальное, инверсное, мигание, с подчеркиванием, с повышенной яркостью).

*Цветной дисплей.* Цветное изображение на экране формируется с помощью трех электронных пушек для основных цветов — зеленого, красного, синего. Экран этих дисплеев имеет мозаичное покрытие люминофорами, вызывающими свечение разного цвета. Электронные лучи, сформированные электронными пушками, пройдя через регулирующую яркость изображения теневою маску, возбуждают свечение триад цветных точек [20]. Цветное изображение как наиболее наглядное широко применяется в графических дисплеях.

*Графические дисплеи* используют точечное задание изображений и растровое их воспроизведение. При высокой разрешающей способности точечное представление позволяет воспроизводить изображение с высокой точностью. Рабочее поле экрана размечается невидимой сеткой, горизонтальные линии которой соответствуют строкам телевизионной развертки. Память регенерации, которая в этом случае имеет значительно большую емкость, чем в алфавитно-цифровых дисплеях, должна хранить сведения (градиация яркости или задания цвета) для всех узлов невидимой сетки — точечных элементов изображения. Последние носят название *пэлов* или *пикселей* [20, 21].

Используются *монохромные* и *цветные графические дисплеи*. В монохромных дисплеях для повышения качества изображения имеется возможность варьирования градаций яркости для каждого пикселя. Еще более эффективным является цветное представление графического изображения с заданием цвета для каждого пикселя. Недостатком точечных дисплеев являются большие затраты времени на регенерацию изображений.

Рисунок на экране может быть выполнен пользователем путем перемещения курсора по экрану или с помощью светового пера, а также устройства «мышь» (см. § 5.5.4). При этом выбор для отдельных элементов изображения градации яркости или цвета производится по соответствующему меню на экране или задается с клавиатуры. В процессе построения изображения информация о состоянии всех его точечных элементов запоминается в памяти регенерации и может быть передана в память ЭВМ. Процесс регенерации сопровождает всю процедуру формирования рисунка на экране.

Изображение (в том числе цветное) на экране может быть сформировано на основании информации, содержащейся в *файле графического изображения*, при этом необходимы соответствующие преобразования данных, что заставляет включать в состав устройства управления — *адаптер цветного дисплея* — отдельный микропроцессор (*дисплейный процессор*), выполняющий собственные дисплейные программы, в результате чего возникает изображение на экране.

Более сложные дисплейные процессоры используются в *векторных дисплеях*, в которых изображение образуется из отдельных отрезков прямых (векторов), которые задаются координатами (абсолютными или относительными) начала и конца вектора и отображаются на экране в результате реализации дисплейным процессором специальной процедуры построения вектора. При этом ЭВМ освобождается от необходимости расчета координат всех точек, из которых формируется изображение вектора [21].

В составе ПК Единой системы ЭВМ используются графические дисплеи, имеющие следующие характеристики.

*Монохромный графический дисплей*: диагональ экрана 31 см; максимальная разрешающая способность  $640 \times 200$  точек; число градаций яркости 16; емкость памяти регенерации 16 Кбайт.

*Цветной графический дисплей*: диагональ экрана 32 см; максимальная разрешающая способность  $640 \times 200$  точек; число цветов 16; емкость памяти регенерации 16 Кбайт.

Оба дисплея могут работать как алфавитно-цифровые (25 строк по 80 знаков в строке).

Развитие применяемых в ПК графических дисплеев идет по пути увеличения разрешающей способности, числа цветов и связанного с этим увеличением емкости памяти регенерации, входящей в состав адаптера дисплея. Так, улучшенный графический адаптер цветного графического дисплея EGA для ПК IBM PC имеет емкость памяти регенерации 256 Кбайт, позволяет при построении изображения одновременно использовать 16 из общего набора 64 цветов и работать с разрешением  $640 \times 350$  пикселей. Для персональных систем PS/2 (см. гл. 10) разработаны векторный дисплей с адаптером MCGA<sup>1</sup>, работающий при разрешении  $320 \times 200$  пикселей, с использованием 256 из общего набора 264 144 цветов, а в монохромном режиме с разрешением  $640 \times 480$  пикселей, а также «Видеографический дисплей» (VGA), имеющий  $640 \times 480$  пикселей при работе с 16 цветами, и, наконец, дисплейный адаптер с улучшенными функциями (AFDA)<sup>2</sup>, обе-

---

<sup>1</sup> MCGA — Multi Colour Graphics Array.

<sup>2</sup> AFDA — Advanced Functions Display Adaptor.



спечивающий разрешение  $1024 \times 768$  пикселей при работе с цветным дисплеем IBM 8514.

Стремление уменьшить габаритные размеры и потребляемую мощность дисплеев, работающих в составе ПК, стимулировали исследования новых подходов для создания малогабаритных *плоских дисплеев* с небольшой мощностью потребления. Такие дисплеи создаются на основе жидких кристаллов, плазменных элементов, электролюминесцентных приборов, плоских ЭЛТ [21].

#### ***5.5.4. Средства управления изображением на экране дисплея персонального компьютера***

Одним из важнейших средств взаимодействия пользователя с экраном дисплея при выборе по представленному на экране «меню», редактировании текста, формировании и управлении изображением является перемещаемый по экрану курсор.

На клавиатуре ПК имеются специальные клавиши, управляющие перемещением курсора. С помощью этих клавиш курсор подводится к соответствующей строке меню (выбор по меню нужной информации, процедуры, режима работы и т. п.), к нужному знакоместу при редактировании (замена, исключение, вставка символа и т. п.).

Более удобны для управления курсором, особенно при работе с изображениями на экране графического дисплея, а также для компьютерных игр специальные манипуляторы, например мышь, джойстик и другие, занимающие заметное место среди периферийных устройств ПК и поддерживаемые входящими в его состав специальными адаптерами и соответствующими обслуживающими программами [20].

*Мышь* (или *колобок*) обычно оформлена в виде небольшой коробочки с несколькими клавишами (рис. 5.16, а), подключаемой к ПК (точнее, к своему адаптеру) гибким кабелем. Пользователь перемещает мышь по поверхности стола, а курсор при этом соответствующим образом движется по экрану. Нажатием клавиш устанавливают начало и конец движения курсора, инициируют операцию редактирования текста, производят выбор по меню.

Устройство мыши поясняет рис. 5.16, б. В прорези в дне коробочки находится шарик, который при ее перемещении по столу вращает диски, расположенные во взаимно перпендикулярных плоскостях. Диски снабжены прорезями, по которым

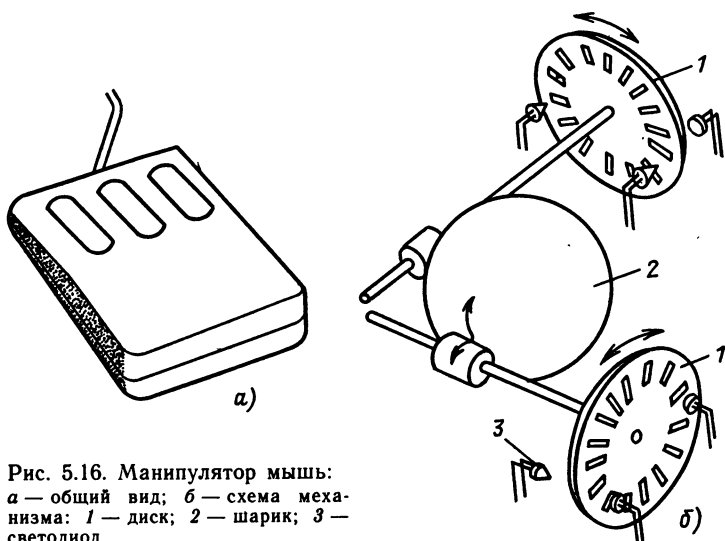


Рис. 5.16. Манипулятор мышь:  
а — общий вид; б — схема механизма: 1 — диск; 2 — шарик; 3 — светодиод

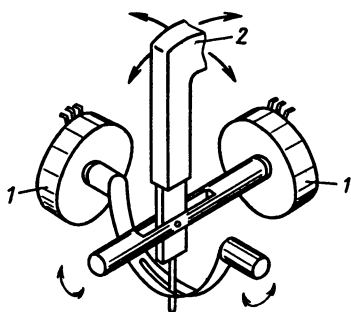


Рис. 5.17. Джойстик:  
1 — потенциометр; 2 — стержень-ручка

фотооптические датчики (пары светодиод — фотодиод) определяют направление и угол поворота дисков.

Джойстик (рис. 5.17) является особенно удобным для компьютерных игр манипулятором, управляющим перемещением по экрану курсора или другого заданного им некоторого графического образа.

Пользователь может повернуть ручку джойстика в любом направлении, перемещая при этом контакты потенциометров, движущиеся во взаимно перпендикулярных плоскостях. Изменяющиеся при этом снимаемые с потенциометров напряжения используются для задания курсору или некоторому графическому образу соответствующего перемещения.

В некоторых конструкциях дисплеев, предназначенных для работы в составе ПК, реализован *чувствительный экран*, который позволяет осуществлять задание режима, команды, выбор по меню нужной информации простым прикосновением пальца к соответствующему месту экрана. При этом часто используется эффект изменения емкости при прикосновении к обкладке конденсатора. На экране дисплея крепится дополнительный прозрачный стеклянный экран, покрытый проводящим слоем, разделенным на несколько десятков сегментов. Каждый сегмент образует конденсаторный элемент, емкость которого контролируется специальной схемой, реагирующей на изменение емкости при прикосновении. Таким образом идентифицируется место, к которому пользователь прикоснулся пальцем.

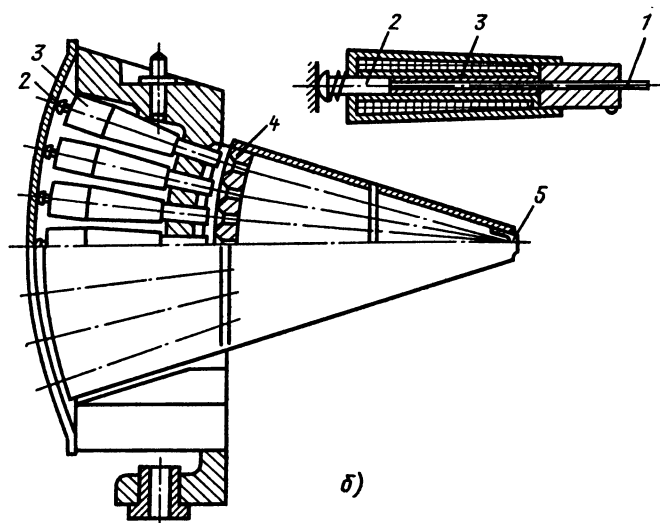
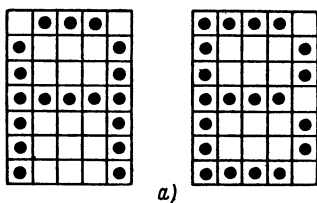
### **5.5.5. Печатающие устройства (принтеры)**

Широкое распространение ПК, в первую очередь высокопроизводительных профессиональных ПК, стимулировало развитие работ по созданию высокоэффективных печатающих устройств (ПЧУ), отличающихся дешевизной, малыми габаритными размерами, небольшой потребляемой мощностью, малыми шумами, возможностью изменения размеров символов и шрифтов, возможностью печати как текстовой информации, так и графических изображений, наличием программного управления цветом печати и, конечно, сравнительно большой скоростью и высоким качеством печати. Понятно, что многие из указанных требований являются взаимно противоречивыми, что привело к разнообразию принципиальных конструктивных решений.

Ограничимся рассмотрением наиболее актуальных для ПК конструкций ПЧУ, к которым следует отнести ударные матричные, ударные типа «ромашка», термопечатающие, а также лазерные устройства высококачественной печати [20, 21].

*Матричные ПЧУ* относятся к *ударным знаковосинтезирующим аппаратам*, основанным на использовании матрицы печатающих элементов (точек или отрезков прямых), из которых при печати образуется контур печатаемого знака. В составе ПК широко применяются последовательные печатающие матричные устройства с матрицей  $5 \times 7$  или  $7 \times 9$  точек. Печатающая головка содержит набор пуансонов — игл (по одной для каждой точки матрицы). Знак фиксируется ударом соответствующего набора игл через красящую ленту по бумаге (рис. 5.18, а). На рис. 5.18, б показан один из вариантов механизма матричной печатающей головки (Dago-1156, ГДР). Головка содержит 35 игл 1, каждая из которых приводится в движение якорем 2 своего миниатюрного электромагнита 3. Иглы движутся в от-

*а* — изображение знаков;  
*б* — механизм печатающей головки



версиях направляющей 4 и в отверстиях выходной матрицы 5. Последняя выполняется из антифрикционного материала.

При работе устройства головка перемещается вдоль строки, печатая последовательно знак за знаком, при этом код очередного знака инициирует срабатывание соответствующей комбинации электромагнитов. Современные матричные ПЧУ печатают при прямом и обратном движениях головки, что обеспечивается наличием в составе устройства соответствующего буферного регистра. Набор печатаемых знаков (алфавит, цифры, другие знаки, а также элементы графических изображений) и их начертание (шрифты) могут видоизменяться программным путем. Также программно устанавливаются размер шрифта и соответственно число знаков в строке, интервал между строками, число строк на странице. Типичные характеристики матричных ПЧУ: двунаправленная печать; число знаков в строке (при сжатом формате шрифта 132, при стандартном шрифте 80, при крупном шрифте 66); скорость печати до 160 знаков/с.

*Ударное последовательно печатающее устройство типа «ромашка»* (рис. 5.19) обеспечивает по сравнению с матричными ПЧУ более высокое качество печати. Печатающая головка («ромашка») представляет собой пластмассовый диск 1, к которому прикреплены гибкие лепестки с литерами 2. При печати шаговый двигатель, поворачивая «ромашку», устанавливает в рабочую позицию нужную литеру, которую затем управляемый электромагнитом молоточек (отсутствует на рисунке) прижимает через красящую ленту 3 к бумаге 4 и валику 5. Печать производится при прямом и обратном движениях каретки с бумагой. Скорость печати 40 знаков/с, число печатаемых знаков около 100. Головки — съемные, и переход на работу с другим шрифтом (алфавитом) производится заменой самой «ромашки». Печатающие устройства типа «ромашка» являются сравнительно малозумными [21].

*Термопринтеры* — дешевые, малогабаритные, малозумные устройства, но со сравнительно невысоким качеством печати, находят применение в простых (бытовых) ПК. В них используется воздействие на бумагу, покрытую термочувствительным слоем, находящихся в регистрирующей головке полупроводниковых термоэлементов, и нагреваются проходящим через них током.

*Лазерный принтер* представляет собой устройство, работающее с высокой скоростью, а главное, с очень высоким качеством печати как текста, так и изображений, причем с возможностью программного задания параметров печати текста и формирования изображений (форма шрифтов, расположение текстов, за-

дания разрешающей способности — число точек на 1 см и др.). Качество печати лазерного принтера не уступает лучшим образцам типографской печати. Лазерные принтеры используются совместно с профессиональными ПК, причем часто из-за сравнительно высокой стоимости этих принтеров одно такое устройство обслуживает несколько ПК в режиме коллективного пользования. Связка профессиональный компьютер — лазерный принтер, составляя основу так называемых настольных издательств, позволяет пользователю готовить хорошо оформленные документы, информационные материалы, макеты изданий и т. п.

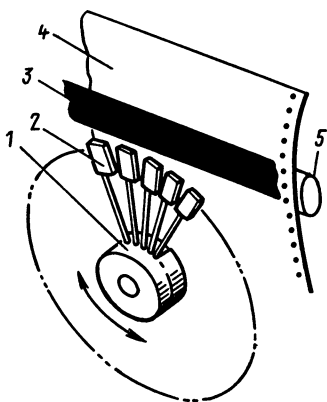


Рис. 5.19. Печатающее устройство типа «ромашка»

Принцип работы лазерного принтера поясняет рис. 5.20 [20]. Основными узлами устройства являются вращающийся цилиндр 1, покрытый фоточувствительным слоем, лазер 2 и развертывающая его луч система, состоящая из вращающегося зеркального шестигранника 3, фокусирующих линз 4 и плоского зеркала 5. Луч лазера 6 перемещается вдоль образующей цилиндра, и лазерные импульсы в соответствии с хранящейся в памяти принтера (обычно емкостью до 1 Мбайт) печатаемой информацией оставляют в соответствующих местах точечные заряды в фоточувствительном слое. За один оборот цилиндра на его поверхности образуются точечные статические заряды, в совокупности отображающие печатаемую страницу. При вращении цилиндр проходит вблизи кармана с мелкодисперсным красителем (тонером). Частицы тонера прилипают к несущим статический заряд точкам поверхности цилиндра. Цилиндр проходит вблизи поступающей из бункера страницы (страница заземлена через металлическую направляющую), на которую переходят частицы тонера, формируя на ней соответствующее изображение. Затем страница подается в нагревательную камеру, в которой происходит закрепление изображения на бумаге.

Лазерные принтеры печатают со скоростью около 10 страниц/мин, с разрешающей способностью 10—15 точек/мм и более. Без замены цилиндра удастся печатать до 10 тыс. страниц.

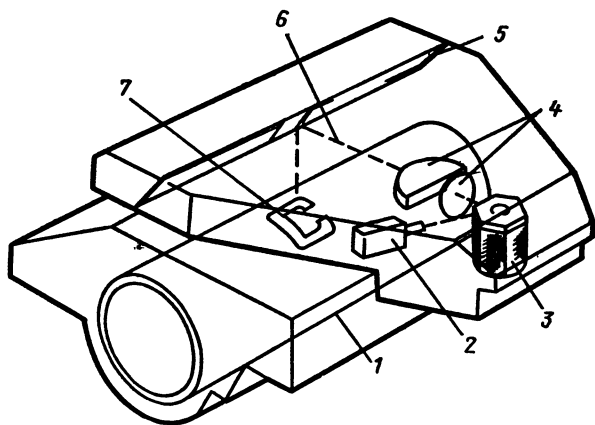


Рис. 5.20. Лазерный принтер:

1 — цилиндр; 2 — маломощный лазер; 3 — зеркальный шестигранник; 4 — фокусирующие линзы; 5 — плоское зеркало; 6 — луч лазера; 7 — электрический заряд на поверхности цилиндра

### **5.5.6. Воспроизведение звуков и музыки в ПК**

Большинство персональных компьютеров имеют встроенные громкоговорители, которые предназначены для выдачи программно задаваемых звуковых сигналов, например от таймера («будильник»), или указывающих на исправность или неисправность при программно управляемом диагностировании оборудования ПК, а также для исполнения запрограммированных музыкальных сообщений.

Программист может завершение отдельных участков своей программы или отдельные переходы в ней обозначить (с помощью соответствующих подпрограмм) исполнением определенных звуковых эффектов или музыки.

На языке БЕЙСИК звуковые эффекты и музыкальные сообщения программируются при помощи операторов соответственно **SOUND** и **PLAY**.

Оператор **SOUND** содержит два числа, определяющих частоту звука (в диапазоне 37—32 767 Гц для ПК IBM PC) и длительность звучания (задается числом импульсов сигналов времени, имеющих частоту 18,2 имп./с, и может составлять от долей секунды до получаса). Программирование мелодий производится при помощи оператора **PLAY**, при этом используется специальный музыкальный язык, содержащий подкоманды, задающие ноты, октавы, длительности звучания нот, исполнение (обычное, легато, стаккато) и другие параметры.

#### **Контрольные вопросы**

1. В чем различия в назначении и характеристиках основной (оперативной) и внешней памяти?
2. Почему ЗУ на магнитных дисках и магнитных лентах относят к ЗУ соответственно прямого и последовательного доступа?
3. Каким образом обеспечивается самосинхронизация считываемой информации при методе записи БВН-1?
4. Какие «интеллектуальные» функции выполняет микропроцессор (микроконтроллер) клавиатуры персонального компьютера?
5. В чем различие алфавитно-цифрового и графического дисплеев?
6. Как в цветном дисплее производится управление цветом?
7. Какие Вы знаете устройства управления изображением на экране дисплея и каков принцип их действия?

## ЯЗЫК МИКРООПЕРАЦИЙ

### 6.1. Декомпозиция вычислительного устройства на операционный и управляющий блоки.

Принцип акад. В. М. Глушкова

Как показал акад. В. М. Глушков, в любом устройстве обработки цифровой информации можно выделить операционный и управляющий блоки [15]. Такой подход упрощает проектирование, а также облегчает понимание процесса функционирования вычислительного устройства. Декомпозиция цифрового вычислительного устройства поясняется на рис. 6.1.

*Операционный блок* состоит из регистров, сумматоров и других узлов, производящих прием из внешней среды и хранение кодов слов, их преобразование и выдачу во внешнюю среду результата преобразования, а также выдачу в управляющий блок и внешнюю среду *оповещающих сигналов*, принадлежащих к множеству

$$U = \{u_1, u_2, \dots, u_n\},$$

о знаках и особых значениях операндов, их отдельных разрядов, особых значениях промежуточных и конечных результатов операции (например, равенство нулю результата операции и др.).

Процесс функционирования во времени устройства обработки цифровой информации состоит из последовательности тактовых интервалов, в которых операционный блок производит определенные элементарные операции преобразования слов. Операционный блок выполняет некоторый набор элементарных преобразований информации, например таких, как передача слова из одного в другой, взятие обратного кода, сдвиг и др. Выполнение этих элементарных операций инициируется поступлением в операционный блок соответствующих управляющих сигналов из некоторого множества сигналов

$$V = \{v_1, v_2, \dots, v_m\}.$$

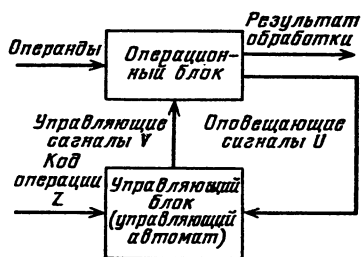


Рис. 6.1. Операционный и управляющий блоки цифрового устройства



Элементарная функциональная операция, выполняемая за один тактовый интервал и приводимая в действие управляющим сигналом, называется микрооперацией.

В некоторые такты могут поступать несколько управляющих сигналов, вызывая параллельное во времени выполнение нескольких микроопераций. Такая совокупность микроопераций называется *микрокомандой*. В частности, микрокоманда может состоять из одной микрооперации.

*Управляющий блок (или управляющий автомат)* вырабатывает распределенную во времени последовательность управляющих сигналов

$$v_{i1}, v_{i2}, \dots, v_{in} (v_{ij} \in V),$$

порождающих в операционном блоке нужную последовательность микроопераций.

Последовательность управляющих сигналов определяется сигналами  $Z$  кода операции, поступающими в управляющий блок извне, и сигналами  $V$ , зависящими от операндов и промежуточных результатов преобразований.

Операционный блок задается его структурой, т. е. составом узлов и связями между ними, и выполняемым операционным блоком набором микроопераций.

Последовательность микрокоманд, обеспечивающая выполнение данной операции (например, операции нормализации числа с плавающей точкой), называется *микропрограммой* данной операции.

Функционирование вычислительного устройства может быть описано совокупностью реализуемых в нем микропрограмм. Это в ряде случаев удобный, хотя и не единственно возможный способ описания цифровых устройств.

## 6.2. Иерархия языков описания вычислительных устройств

Вычислительные машины и системы вместе с их программным обеспечением можно отнести к наиболее сложным системам, созданным человеком. Их сложность определяется многочисленностью входящих в их состав элементов, большим числом связей между элементами и сложностью осуществляемых законов функционирования и преобразования информации. Как и в других сложных системах, рассмотрение и описание структуры и функционирования ЭВМ и системы в зависимости от преследуемой цели производятся с различной степенью детализации, определяемой уровнем, на котором анализируется процесс функционирования. Каждому уровню рассмотрения соответствуют определенные средства описания, которые, по существу, являются языком описания вычислительного устройства для данного уровня представления.

Сказанное можно проиллюстрировать на материале предыдущих глав. Например, работу элемента, изображенного на рис. 3.5, а, можно рассматривать с точки зрения процессов изменения электрических токов и напряжений в отдельных точках схемы. В этом случае формальным средством описания или языком описания являются дифференциальные уравнения для токов и напряжений в цепях схемы, в основе которых лежат законы физики полупроводников и законы электротехники. На таком уровне описания рассматриваются рабочие процессы электронных схем. Однако если такой подход применить к более сложному устройству, хотя бы, например, к комбинационному параллельному сумматору, то описание окажется весьма сложным и по нему трудно представить себе производимое схемой преобразование информации. Чтобы сделать это нагляднее, необходимо отвлечься от физических процессов, происходящих в схеме сумматора, и рассмотреть его работу на другом, более обобщенном уровне — на уровне комбинационных схем и использовать в качестве языка описания булевы функции. В этом случае в качестве входных и выходных переменных рассматриваются не электрические напряжения и токи в соответствующих точках схемы, а булевы переменные, принимающие всего два значения: 0 и 1.

Обратимся теперь к блокам, содержащим элементы памяти, например регистры, и выполняющим последовательности элементарных актов преобразования информации, называемых микрооперациями. Такие блоки в предыдущем параграфе названы операционными. На уровне операционных блоков булевы функции уже недостаточны для построения описания.

Может показаться, что в этом случае, а тем более при рассмотрении более сложных образований — функциональных устройств, содержащих блоки памяти, несколько операционных и управляющих блоков или совокупностей взаимодействующих функциональных устройств, можно ограничиться словесным (вербальным) описанием процесса функционирования. Однако легко убедиться, что такой способ описания является и недостаточно лаконичным, и в то же время недостаточно строгим и точным, чтобы обеспечить однозначное представление о процессе функционирования устройства или машины.

Потребность в формализованных средствах описания структур и функционирования цифровых устройств машин определяется не только и не столько задачами обучения, сколько потребностями современной методологии проектирования ЭВМ. В последней существенное место занимают моделирование на ЭВМ проектных решений в целях их проверки и оптимизации, автоматизация с помощью ЭВМ конструирования вычислительных устройств и их отдельных узлов и синтез контрольных и диагностических тестов и т. д. Для этого необходимы формализованные описания проектируемых устройств. Без формализованных описаний нельзя достигнуть соответствующего взаимопонимания между объединенными общим проектом разработчиками отдельных функциональных устройств.

Для формализованного описания устройств вычислительной техники используют различные «языки описания» в зависимости от необходимой степени детализации структуры и процесса функционирования. Различным уровням рассмотрения вычислительного устройства соответствует определенная иерархия языков описания (табл. 6.1).

Отметим, что нет четкой границы между уровнями рассмотрения, на которых можно использовать те или иные языки описания. Например, операционные блоки можно описать средствами языка описания функционирования вычислительных устройств (язык APL), но такое описание окажется во многих случаях недостаточно детализированным и наглядным. Работу устройства ЭВМ можно описать на языке микроопераций,

**Т а б л и ц а 6.1. Иерархия языков формализованного описания  
вычислительных устройств**

Уровень рассмотрения	Язык формализованного описания
I. Электронные схемы'	Дифференциальные уравнения для токов и напряжений в цепях электрических схем
II. Комбинационные логические схемы	Аппарат теории булевых функций
III. Операционные узлы, узлы памяти, управляющие автоматы	Язык микроопераций
IV. Устройства вычислительной машины	Языки МОДИС-ВЕС, APL [76], ООС-2 и др.
V. Функционирование вычислительной машины	Языки машинных команд
VI. Вычислительный процесс	Алгоритмические языки (ФОРТРАН, ПЛ/1, ПАСКАЛЬ, Си и др.)
VII. Функционирование вычислительной системы в условиях действия случайных факторов	Язык моделирования дискретных стохастических систем (GPSS и др.)

но это описание окажется слишком детализированным и перегруженным подробностями, что будет затруднять понимание процесса функционирования.

Подобным образом реализуемый ЭВМ вычислительный процесс можно описать и на алгоритмическом языке высокого уровня (например, на языке ПЛ/1) и на языке команд ЭВМ. В последнем случае вычислительный процесс будет описан более подробно, все детали его выполнения однозначно определены. Такое описание пригодно для задания вычислительного процесса ЭВМ, но из-за слишком большой степени детализации и обилия подробностей воспринимается человеком несравненно хуже, чем описание вычислительного процесса на алгоритмическом языке высокого уровня.

Переход с одного уровня рассмотрения на другой, соответствующий в табл. 6.1 движению снизу вверх, сопровождается увеличением степени детализации описания. Движение сверху вниз означает переход к описанию, позволяющему лучше понять процесс функционирования.

До сих пор речь шла о так называемых детерминированных объектах (устройствах, машинах, системах), при рассмотрении которых не учитываются действия случайных факторов. Учет случайных факторов (сбоев, отказов, потоков запросов на обработку программ и др.), необходимость в котором возрастает по мере перехода к рассмотрению более сложных вычислительных устройств и комплексов, требует привлечения методов описания и анализа дискретных стохастических систем. Для анализа подобных систем широко используется статистическое моделирование на ЭВМ процесса функционирования. Поэтому в качестве средства их описания могут служить языки моделирования дискретных стохастических систем (GPSS и др.)

### 6.3. Язык микроопераций

Язык микроопераций (ЯМ) предназначен для описания цифровых устройств, функционирование которых рассматривается на уровне регистров. Поэтому иногда такой язык называют регистровым или языком регистровых передач. Он имеет простые и наглядные средства описания слов и регистров, массивов и памяти, элементов и частей слов и массивов и соответственно элементов регистров и памяти, операций передачи слов и частей слов и др.

*Описание слов, регистров и шин.* Описание слова (числа, логического кода и др.) содержит его название — идентификатор и разрядный указатель. Идентификатором служит произвольная последовательность букв и цифр, начинающаяся с буквы. Разрядный указатель состоит из номеров старшего и младшего разрядов слова, разделенных знаком  $\div$ . Указатель заключается в квадратные скобки. Например, слово-число

$$X_{15} = \alpha_0 \alpha_1 \dots \alpha_n,$$

где  $\alpha_i$  — двоичные разряды, можно предоставить в виде

$$X_{15} [0 \div n].$$

Разрядный указатель слова может опускаться, если это не вызывает недоумений, в частности, если ранее слово уже полностью описано. Одноразрядное слово описывается только идентификатором без разрядного указателя.

Аналогичным образом описание регистра состоит из названия (идентификатора) регистра и разрядного указателя. Например, описание регистра команды на рис. 6.2 имеет вид  $PzK [0 \div 31]$ , а его отдельных частей (подрегистров) и соответственно полей команды

$PzK [0 \div 7]$  или  $PzK [KOn]$ ;

$PzK [20 \div 31]$  или  $PzK [A_2]$ ,

т. е. в разрядном указателе подрегистра можно указать идентификатор подрегистра (поля слова). Значение некоторого, например седьмого, разряда регистра  $PzK$  выделяется записью  $PzK [7]$ .

Совокупность линий (цепей), предназначенных для передачи слова, или, в более общем случае, кодов и сигналов, объединенных общим функциональным назначением, называется шиной. Шина, по которой в цифровое устройство извне поступает или во внешнюю среду выдается слово, описывается, как и регистр, идентификатором шины и разрядным указателем. Например, описание шины, по которой поступает 32-разрядная команда, обозначается  $ШК [0 \div 31]$ .

*Описание массива данных и памяти.* Описание массива, состоящего из слов одинаковой длины, и памяти содержит их название — идентификатор (например, ОП2 для модуля 2 оперативной памяти) и в квадратных скобках — наименьший и наибольший номера слов или нижнюю и верхнюю границы массива (наименьший и наибольший номера ячеек памяти), а также порядок нумерации разрядов в словах. Пример описания модуля памяти ОП2 (массива), содержащего  $r$   $n$ -разрядных ячеек (слов): ОП2  $[0 \div r-1, 0 \div n-1]$ . В таком случае  $j$ -я  $n$ -разрядная ячейка памяти ( $j$ -е слово) и  $k$ -й разряд памяти (столбец массива) представляются записями соответственно ОП2  $[j, 0 \div n-1]$  и ОП2  $[0 \div r-1, k]$ .

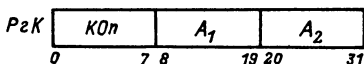


Рис. 6.2. К описанию регистра

**Описание микроопераций.** Микрооперация осуществляет некоторое преобразование над данными. Это преобразование может быть логическим, т. е. выполняемым над операндом или двумя операндами поразрядно, арифметическим или функциональным. Например, это может быть установка некоторого слова в регистре, передача, взятие обратного кода, операции И и ИЛИ над двумя операндами, составление слова, сдвиг, операция счета, операция сложения и вычитания, сравнение на равенство и др. Микрооперации могут быть одноместными или многоместными, в них участвует соответствующее число операндов. Микрооперация описывается микрооператором и может сопровождаться меткой — идентификатором микрооперации. Характерный вид описания двухместной микрооперации

$$\begin{array}{c} \text{Метка} \quad G: \quad \underbrace{PzA [k \div k + l] := PzB [m \div m + l] * PzC [n \div n + l]}_{\text{Микрооператор}} \end{array} \quad (6.1)$$

Формула микрооператора

Метка (идентификатор) отделяется двоеточием от микрооператора. Представление микрооператора основано на использовании «операции присваивания», обозначаемой знаком  $:=$ . Выражение, стоящее справа от этого знака (правая часть микрооператора), называется «формулой микрооператора». Формула определяет преобразование [в (6.1) обозначено знаком  $*$ ], производимое микрооперацией, и участвующие в нем операнды, или, точнее, их местоположение. Слева от знака присваивания (в левой части микрооператора) указывается регистр (или его часть), в который передается результат преобразования, описанного формулой микрооператора. Если в левой части микрооператора указана некоторая часть регистра (подрегистр), то после выполнения микрооперации остальные разряды регистра сохраняют прежнее значение.

Действие микрооператора состоит в том, что в конце такта выполнения микрооперации на регистре (шине и др.), описанном в левой части микрооператора, устанавливается слово, полученное в результате указанных в формуле микрооператора преобразований над значениями операндов в начале такта. В частном случае, если микрооперация состоит в передаче слова, формула микрооператора содержит лишь описание слова, точнее, регистра (или части регистра), из которого происходит передача. Например, микрооперацию приема адреса второго операнда  $A_2$  из регистра команды  $PzK$  в регистр адреса  $PzA$  (рис. 6.3) можно пред-

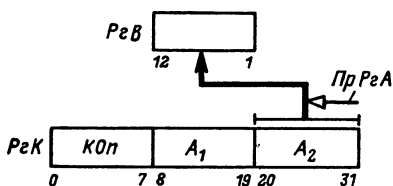


Рис. 6.3. Микрооперация: прием адреса  $A_2$  из регистра  $PzK$  в регистр  $PzA$

ставить в виде

$$ПрР_2A : P_2A [12 \div 1] := P_2K [20 \div 31];$$

или в более простой форме

$$ПрР_2A : P_2A := P_2K [A_2];$$

где  $ПрР_2A$  — метка микрооперации *Прием* в  $P_2A$  адреса  $A_2$  из  $P_2K$ . Управляющий сигнал, вызывающий выполнение данной микрооперации, показан условно на рис. 6.3 тонкой стрелкой.

Управляющий сигнал микрооперации, рассматриваемый как переменная, принимает лишь два значения: 1 — микрооперация возбуждается, 0 — микрооперация не возбуждается. Один управляющий сигнал может инициировать выполнение в одном такте нескольких микрооператоров (микрокоманд). Тогда последние записываются подряд и отделяются друг от друга запятыми. Точка с запятой разделяет микрооператоры, выполняющиеся в разных тактах.

Пусть, например, на рис. 6.4 прием из счетчика команд  $CчК$  в регистр  $P_2B$  и из регистра  $P_2D$  в  $CчК$  выполняется за один такт, тогда можно записать

$$P_2B := CчК, CчК := P_2D.$$

Рассмотрим еще некоторые примеры микроопераций.

**Приращение счетчика.** В счетчике на рис. 6.5 определены микрооперации: установка в 0 ( $УОСч$ ), прием кода с шины  $ШИВх$  ( $ПрСч$ ) и увеличение содержимого счетчика на 1 ( $+1Сч$ ). Последнюю микрооперацию можно записать в виде

$$Cч := Cч + 1;$$

**Сдвиг.** Различают сдвиги арифметический, логический, циклический. В описаниях микроопераций им соответствуют обозначения  $СдвA$ ,  $СдвЛ$ ,  $СдвЦ$ . Далее за этим обозначением указываются направление сдвига (правый  $П$  или левый  $Л$ ) и в скобках число разрядов, на которое производится сдвиг. Например, описание микрооперации арифметического сдвига содержимого регистра  $C$  вправо на четыре разряда имеет вид

$$P_2C := СдвАП (4) P_2C.$$

При арифметическом сдвиге знаковый разряд не сдвигается. Освобождающиеся при сдвиге разряды заполняются 0. Выдвигающиеся из регистра биты слова теряются.

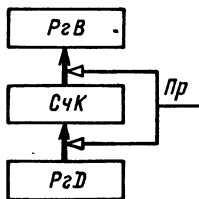


Рис. 6.4. Пример совмещенных во времени микрооператоров

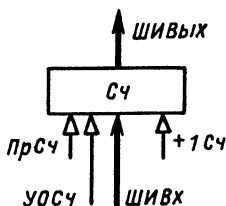


Рис. 6.5. Микрооперации счетчика

При логическом сдвиге сдвигаются все разряды кода, включая знаковый разряд. При циклическом сдвиге крайние разряды регистра соединяются между собой так, что выдвигающиеся из крайнего разряда регистра биты поступают в другой его крайний разряд.

Сдвиг часто производится путем «косой передачи» слова из одного регистра в другой. Микрооперации передачи содержимого  $P_2A$  в  $P_2B$  со сдвигом на  $k$  разрядов вправо или влево обозначаются соответственно

$$P_2B := \Pi(k) P_2A;$$

$$P_2B := \Pi(k) P_2A.$$

**Конкатенация или составление слова.** Пусть, например, в регистр  $B$  должно быть передано слово, отдельно разряды которого соответствуют содержимому некоторых разрядов регистра  $A$ , счетчика  $Cч$ , триггера переполнения  $TзП$  и константе 0, как это показано на рис. 6.6. Описание соответствующей микрооперации имеет вид

$$P_2B [1 \div 16] := P_2A [8 \div 15] \parallel Cч [0 \div 3] \parallel TзП \parallel 000;$$

Вертикальная линия является знаком операции составления слова.

Для описания требуемой последовательности выполнения микроопераций в Ям используется оператор перехода, который для удобства будем называть микрооператором, не связывая его выполнение с выполнением микрооперации.

Микрооператор перехода

идти к  $M$ ;

позволяет задавать в микропрограмме переход к микрооператору с меткой  $M$ .

**Условные микрооператоры.** Часто в микропрограммах в зависимости от того, соблюдается или не соблюдается некоторое условие, должна выполняться та или иная микрооперация, или, что фактически одно и то же, должна выполняться или пропускаться некоторая микрооперация. В этих случаях для записи микрооперации используются условные микрооператоры одного из следующих типов: **если (условие) то микрооператор(ы) иначе микрооператор(ы)**.

Вместо микрооператора в условном микрооператоре может стоять метка, отсылающая к микрооператору, определенному в другом месте микропрограммы. Такой микрооператор называется микрооператором условного перехода.

Для записи микрооперации можно воспользоваться также условным

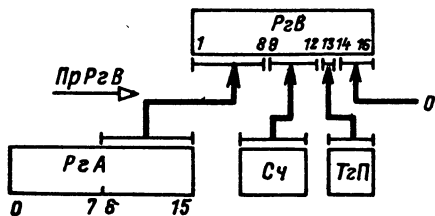


Рис. 6.6. Микрооперация составления (конкатенации) слова

микрооператором вида (для двухместных микроопераций)

$PzA [k \div k + l] : \text{если (условие) то } PzB [m \div m + l] *$

$* PzC [n \div n + l] \text{ иначе } PzD [p \div p + l] * PzE [q \div q \div l];$

Условие можно представить как равенство (или отсутствие равенства) содержимого некоторого регистра (или разряда регистра) определенной величины, например  $Cm = 0$  или  $PzA [0] \neq 0$ , или в форме неравенства ( $Cm > 0$ ), или в виде логической функции двоичных переменных, в качестве которых могут выступать значения определенных разрядов регистров. В последнем случае условие выполняется, если логическая функция принимает значение 1 (истинно).

Запись микрооперации можно сопровождать пояснениями, которые должны при этом выделяться скобками  $<, >$ .

Описание микрокоманды производится аналогично описанию микрооперации и представляет собой метку микрокоманды и разделенную запятыми совокупность микрооператоров, выполняемых в микрокоманде. В общем случае можно говорить о микрокоманде, так как микрооперация является частным случаем микрокоманды, содержащей только одну микрооперацию.

Обычно даже очень простые операции преобразования информации занимают несколько тактов и требуют выполнения определенной последовательности микрокоманд. Например, если на рис. 6.3 прием в  $PzA$  адреса требует предварительной установки  $PzA$  в 0, то прием в  $PzA$  адреса второго операнда  $A$ , из регистра  $PzK$ , будет производиться последовательностью микрокоманд (т. е. микропрограммой)

1-й такт  $Уст0PzA : PzA : = 0;$

2-й такт  $ПрA_2 : PzA : = PzK[A_2];$

Восприятие микропрограммы облегчается, если отдельные микрокоманды или группы микрокоманд, соответствующие определенным функциональным операциям, обозначить «метками операций». Метка операции предшествует описанию первой микрокоманды из ее группы и отделяется от этой микрокоманды знаком  $::$ .

Микропрограмма может быть изображена в виде графа, отдельные вершины которого соответствуют микрокомандам или группам последовательно выполняемых микрокоманд. Безусловные микрокоманды обозначаются прямоугольниками, а условные — ромбами с показом разветвлений. Внутри прямоугольников и ромбов записывают выражения для микрооператоров. Метки, если они используются, проставляют сбоку прямоугольника. Следует отметить, что все метки микропрограммы должны быть различными. Это не позволяет отождествить их с именами управляющих сигналов при наличии в микропрограмме одинаковых микрокоманд. В то же время отождествление (всюду, где оно возможно) позволяет сделать текст микропрограммы более наглядным и содержательным.

В качестве примера на рис. 6.7 приведена представленная в виде графа микропрограмма рабочего цикла памяти с произвольным обращением, словесное описание функционирования которой проводилось в § 4.2.

Как видно из рис. 4.2, в БУП поступают извне управляющие сигналы *Обращение* (сигнал начала рабочего цикла) и *Операция* (1 — считывание, 0 — запись); БУП генерирует в необходимой последовательности управляющие сигналы, инициирующие соответствующие микрооперации: *ПрPzA* — прием адреса с шин *ША* в  $PzA$ . *Считывание* —



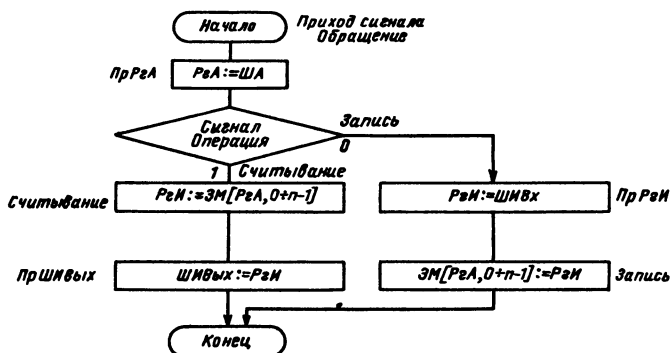


Рис. 6.7. Микропрограмма рабочего цикла памяти с произвольным обращением

открытие усилителей считывания и передача считанного кода в  $РсИ$ ;  $ПрРсИ$  — прием в  $РсИ$  слова с  $ШИВх$ ;  $Запись$  — возбуждение разрядных усилителей записи;  $ПрШИВых$  — выдача слов из  $РсИ$  на  $ШИВых$ .

### Контрольные вопросы

1. В чем отличие микрокоманды от микрооперации?
2. От чего зависит последовательность управляющих сигналов, вырабатываемая управляющим блоком?
3. Для чего нужны формализованные средства описания вычислительных устройств?
4. Для чего предназначен язык микроопераций?
5. В чем состоит действие микрооператора?
6. Как в языке микроопераций описывается выполнение условных действий?

## Глава 7

# ПРИНЦИПЫ ОРГАНИЗАЦИИ АРИФМЕТИЧЕСКО-ЛОГИЧЕСКИХ УСТРОЙСТВ

## 7.1. Общие сведения

Арифметическо-логические устройства (АЛУ) служат для выполнения арифметических и логических преобразований над словами, называемыми в этом случае операндами.

Выполняемые в АЛУ операции можно разделить на следующие группы:

- операции двоичной арифметики для чисел с фиксированной точкой;

- операции двоичной (или шестнадцатеричной) арифметики для чисел с плавающей точкой;

- операции десятичной арифметики;

- операции индексной арифметики (при модификации адресов команд);

- операции специальной арифметики;

- операции над логическими кодами (логические операции);

- операции над алфавитно-цифровыми полями.

Современные ЭВМ общего назначения обычно реализуют операции всех приведенных выше групп, а малые и микроЭВМ, микропроцессоры и специализированные ЭВМ часто не имеют аппаратуры арифметики чисел с плавающей точкой, десятичной арифметики и операций над алфавитно-цифровыми полями. В этом случае эти операции выполняются специальными подпрограммами.

К арифметическим операциям относятся сложение, вычитание, вычитание модулей («короткие операции») и умножение и деление («длинные операции»). Группу логических операций составляют операции дизъюнкция (логическое ИЛИ) и конъюнкция (логическое И) над многоразрядными двоичными словами, сравнение кодов на равенство. Специальные арифметические операции включают в себя нормализацию, арифметический сдвиг (сдвигаются только цифровые разряды, знаковый разряд остается на месте), логический сдвиг (знаковый разряд сдвигается вместе с цифровыми разрядами). Обширна группа операций редактирования алфавитно-цифровой информации.

Можно привести следующую классификацию АЛУ.

По способу действия над операндами АЛУ делятся на последовательные и параллельные. В последовательных АЛУ операнды представляются в последовательном коде, а операции производятся последовательно во времени над их отдельными разрядами. В параллельных АЛУ операнды представляются параллельным кодом и операции совершаются параллельно во времени над всеми разрядами операндов.

По способу представления чисел различают АЛУ:

- 1) для чисел с фиксированной точкой;

- 2) для чисел с плавающей точкой;

- 3) для десятичных чисел.

По характеру использования элементов и узлов АЛУ делятся на блочные и многофункциональные. В блочном АЛУ операции над числами с фиксированной и плавающей точкой, десятичны-

ми числами и алфавитно-цифровыми полями выполняются в отдельных блоках, при этом повышается скорость работы, так как блоки могут параллельно выполнять соответствующие операции, но значительно возрастают затраты оборудования. В многофункциональных АЛУ операции для всех форм представления чисел выполняются одними и теми же схемами, которые коммутируются нужным образом в зависимости от требуемого режима работы.

По своим функциям АЛУ является операционным блоком (см. § 6.1), выполняющим микрооперации, обеспечивающие прием из других устройств (например, памяти) операндов, их преобразование и выдачу результатов преобразования в другие устройства. Арифметическо-логическое устройство управляется управляющим блоком, генерирующим управляющие сигналы, инициирующие выполнение в АЛУ определенных микроопераций. Генерируемая управляющим блоком последовательность сигналов определяется кодом операции команды и оповещающими сигналами.

В заключение приведем характерные параметры для АЛУ современных ЭВМ: длина слова 32 разряда (4 байта), скорость работы — миллионы сложений чисел с фиксированной точкой в секунду.

## **7.2. Структура и микропрограмма АЛУ для сложения и вычитания чисел с фиксированной точкой**

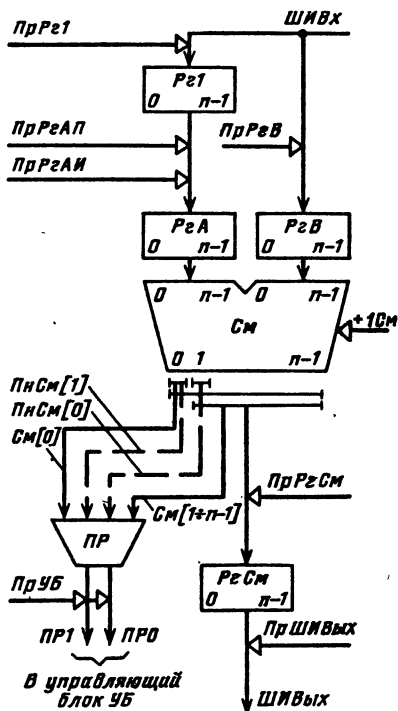
Обычно в АЛУ операция алгебраического сложения сводится к арифметическому сложению кодов чисел путем применения инверсных кодов — дополнительного или обратного для представления чисел. Обратный код имеет два представления нуля ( $+0$ ,  $-0$ ), что затрудняет анализ результата операции. Чаще используется дополнительный код.

Алгоритмы выполнения в АЛУ арифметических операций зависят от того, в каком виде хранятся в памяти ЭВМ числа — в прямом или дополнительном. В последнем случае сокращается время выполнения операции за счет исключения операции преобразования получаемого в АЛУ дополнительного кода отрицательного результата в прямой код, хотя при этом несколько усложняется операция умножения.

*Алгоритм алгебраического сложения двоичных чисел с фиксированной точкой при использовании дополнительного кода для представления чисел.* При выполнении операции сложения производится сложение двоичных кодов, включая разряды знаков.

Если при этом возникает перенос из знакового разряда суммы при отсутствии переноса в этот разряд или перенос в знаковый разряд при отсутствии переноса из разряда знака, то имеется переполнение разрядной сетки соответственно при отрицательной и положительной суммах. Если нет переносов из знакового разряда и в знаковый разряд суммы или есть оба эти переноса, то переполнения нет и при 0 в знаковом разряде сумма положительна, а при 1 в знаковом разряде сумма отрицательна.

На рис. 7.1 представлена упрощенная структурная схема АЛУ для операций сложения и вычитания  $n$ -разрядных (нулевой разряд знаковый) двоичных чисел с фиксированной точкой. Предполагается, что числа хранятся в памяти в дополнительном коде.



В состав АЛУ входят  $n$ -разрядный параллельный комбинационный сумматор  $C_m$ , регистр для сложения и вычитания двоичных чисел с фиксированной точкой сумматора  $PzCm$ , входные регистры сумматора  $PzB$  и  $PzA$ , входной регистр АЛУ  $PzI$ .

Из оперативной памяти по входной информационной шине ШИВх в АЛУ поступают операнды. Операнды размещаются в  $PzB$  (первое слагаемое или уменьшаемое) и  $PzI$  (второе слагаемое или вычитаемое);  $PzI$  соединен с  $PzA$  цепями прямой и инверсной передачи кода. Прямая передача используется при операции алгебраического сложения, а инверсная — при операции вычитания. Результат операции выдается из АЛУ в оперативную память по выходной информационной шине ШИВых.

При выполнении операции в АЛУ помимо результата операции формируется 2-разрядный код признака результата  $PR$ , который принимает следующие значения:

Результат операции	Признак результата	
0	0	0
< 0	0	1
> 0	1	0
Переполнение	1	1

Примем, что код признака результата формируется комбинационной схемой  $ПР$ , на входы которой поступают сигналы, соответствующие значениям всех разрядов сумматора, а также сигналы переносов из знакового разряда  $ПнСм[0]$  и в знаковый из старшего цифрового разряда  $ПнСм[1]$ . Признак переполнения ( $ПР=11$ ) формируется, если булева функция

$$ПнСм[0] \overline{ПнСм[1]} \vee \overline{ПнСм[0]} ПнСм[1] = 1.$$

Признак нулевого значения результата  $ПР=00$  формируется, если

$$\bigwedge_{i=0}^{n-1} \overline{См[i]} = 1.$$

Условия выработки признаков отрицательного и положительного результатов имеют соответственно вид

$$\overline{См[0]} (ПнСм[0] ПнСм[1] \vee \overline{ПнСм[0]} \overline{ПнСм[1]});$$

$$См[0] (ПнСм[0] ПнСм[1] \vee \overline{ПнСм[0]} \overline{ПнСм[1]});$$

При выполнении алгебраического сложения поступившие в АЛУ коды операндов находятся на входных регистрах  $РгВ$  и  $РгА$  сумматора. Код суммы формируется на выходах схемы  $См$  и фиксируется в регистре  $РгСм$ .

Операция алгебраического вычитания

$$Z = X - Y = X + (-Y)$$

может быть сведена к изменению знака вычитаемого  $Y$  и операции алгебраического сложения. Изменению знака соответствует следующая процедура (см. гл. 2): принятый в  $РгI$  код числа передается инверсно в  $РгА$  и при сложении осуществляется подсуммирование 1 в младший разряд сумматора.

Передачи информации в регистрах АЛУ производятся отдельными микрооперациями, иницируемыми показанными на рис. 7.1 управляющими сигналами. Слово из  $РгI$  в  $РгА$  может быть передано в прямом (управляющий сигнал  $ПрРгАП$ ) или в инверсном (управляющий сигнал  $ПрРгАИ$ ) кодах.

Ниже приведено совмещенное описание микропрограммы операций сложения и вычитания на языке ЯМ:

$ПрРгВ : РгВ := ШИВx$  (прием I операнда);

$ПрРгI : РгI := ШИВx$  (прием II операнда);

Прием: если сложение то  $РгА := РгI$  иначе  $РгА := \overline{РгI}$ ;

Сумма: если сложение то  $РгСм := РгА + РгВ$  иначе

$$РгСм := РгА + РгВ + 1;$$

*ПрУБ* : если  $PR=11$  то прерывание иначе

*ПрШИВых*: *ШИВых* :=  $R_2Cm$  (выдача результата);

**конец**

Микрооперация *ПрУБ* состоит в выдаче в управляющий блок кода признака и в формировании запроса прерывания при переполнении разрядной сетки.

### 7.3. Структуры и микропрограмма АЛУ для умножения чисел с фиксированной точкой

В ЭВМ операция умножения чисел с фиксированной точкой с помощью соответствующих алгоритмов сводится к операциям сложения и сдвига.

Произведение двух  $(n-1)$ -разрядных чисел может иметь  $2(n-1)$  значащих разрядов. Поэтому при операции умножения целых чисел необходимо предусмотреть возможность формирования в АЛУ произведения, имеющего двойную по сравнению с сомножителями длину. В ЭВМ, в которых числа с фиксированной точкой являются дробями, младшие  $n-1$  разрядов произведения часто отбрасываются (при отбрасывании может производиться операция округления произведения).

Для выполнения умножения АЛУ должно содержать регистры множимого, множителя и схемы формирования суммы частичных произведений — так называемый сумматор частичных произведений, в котором (рис. 7.2) путем соответствующей организации передач производится последовательное суммирование частичных произведений.

Операция умножения состоит из  $n-1$  [ $n-1$  — число цифровых разрядов множителя] циклов. В каждом цикле анализируется очередная цифра множителя, и если это 1, то к сумме частичных произведений прибавляется множимое, в противном случае прибавление не происходит. Цикл завершается сдвигом множимого относительно суммы частичных произведений либо сдвигом суммы частичных произведений относительно неподвижного множимого.

В зависимости от способа формирования суммы частичных произведений различают четыре основных метода выполнения умножения и соответственно четыре структуры АЛУ для этой операции (рис. 7.3). Для определенности сна-

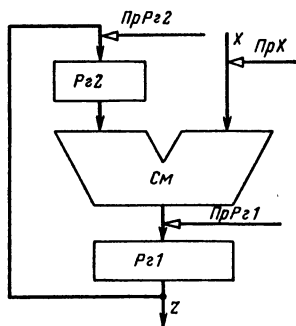


Рис. 7.2. Сумматор частичных произведений

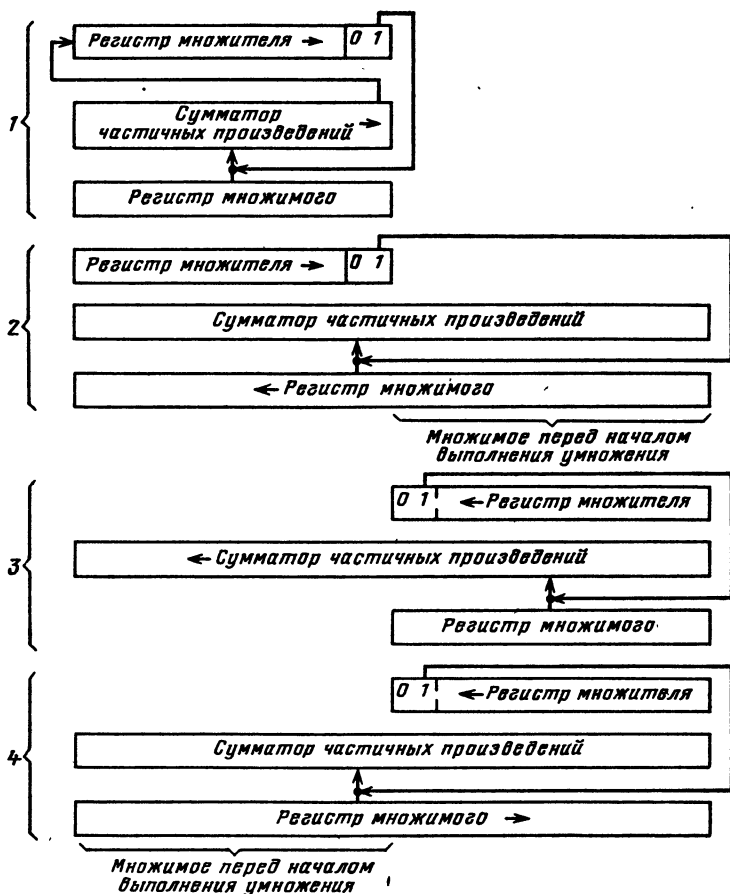


Рис. 7.3. Методы выполнения умножения

чала будем считать, что оба сомножителя — положительные числа.

1. Умножение, начиная с младших разрядов множителя, со сдвигом суммы частичных произведений вправо и при неподвижном множимом.

Регистр множителя и сумматор частичных произведений при этом должны иметь цепи сдвига вправо. Регистр множимого может не иметь цепей сдвига.

Последовательность действий в каждом цикле выполнения умножения определяется младшим разрядом регистра множителя, куда последовательно одна за другой поступают цифры множителя.

Поскольку по мере сдвига множителя вправо старшие разряды регистра множителя освобождаются, он может быть использован для хранения младших разрядов произведения, поступающих из младшего разряда сумматора частичных произведений по мере выполнения умножения. Для этого при выполнении сдвига младший разряд регистра сумматора частичных произведений соединяется со старшим разрядом регистра множителя. После выполнения умножения старшие разряды произведения находятся в регистре сумматора, младшие — в регистре множителя.

При данном методе умножения все три регистра имеют одинаковую длину, равную числу разрядов сомножителей. Этот метод умножения нашел наибольшее применение в ЭВМ.

2. Умножение, начиная с младших разрядов множителя, при сдвиге множимого влево и неподвижной сумме частичных произведений.

Регистр множителя при этом должен иметь цепи сдвига вправо, регистр множимого — цепи сдвига влево, а сумматор частичных произведений не содержит цепей сдвига.

Последовательность действий определяется, как и в первом варианте, младшим разрядом регистра множителя.

При этом методе регистр множимого и сумматор частичных произведений должны иметь двойную длину. Этот метод требует больше оборудования, но никаких преимуществ не дает, и поэтому применение его нецелесообразно.

3. Умножение, начиная со старших разрядов множителя, при сдвиге суммы частичных произведений влево и неподвижном множимом. Регистр множителя и сумматор частичных произведений должны иметь цепи сдвига влево. Регистр множимого не имеет цепей сдвига.

Последовательность действий в каждом цикле выполнения умножения определяется старшим разрядом регистра множителя.

При этом методе сумматор частичных произведений должен иметь двойную длину. Данный метод требует дополнительного по сравнению с первым методом оборудования. Несмотря на это, он применяется в некоторых АЛУ, так как позволяет без дополнительных цепей сдвига выполнять и деление.

Для выполнения операций деления в АЛУ, реализующем первый метод умножения, необходимы дополнительные цепи сдвига влево в регистре множимого (частного) и в сумматоре частичных произведений (разностей).

4. Умножение, начиная со старших разрядов множителя, при сдвиге вправо множимого и неподвижной сумме частичных произведений.



Регистр множителя должен иметь цепи сдвига влево, регистр множимого — цепи сдвига вправо. Сумматор частичных произведений не имеет цепей сдвига. Последовательность действий на каждом шаге умножения определяется старшим разрядом регистра множителя.

При этом методе умножения и регистр множимого, и сумматор частичных произведений должны иметь двойную длину. Однако, как и третий метод, он не требует дополнительных цепей сдвига для выполнения деления.

При четвертом методе, в котором сумма частичных произведений неподвижна, можно совмещать во времени операции сдвига и сложения и за этот счет увеличить быстродействие АЛУ при выполнении умножения (деления).

Если необходимо образование произведения двойной длины, например, при операциях с целыми числами, наиболее экономичным является первый из рассмотренных методов умножения, так как он позволяет использовать все регистры одинарной длины.

Если в результате умножения достаточно иметь произведение одинарной длины, то целесообразно использовать либо первый, либо четвертый метод умножения. При использовании первого метода требуется введение дополнительных цепей сдвига для реализации деления, а при использовании четвертого метода необходимо удлинение сумматора. Выбор одного из этих методов умножения определяется соотношением затрат оборудования на реализацию цепей сдвига и дополнительных разрядов сумматора.

При образовании произведений одинарной длины простое отбрасывание младших разрядов вносит погрешность, которая будет накапливаться, так как произведение будет всегда вычисляться с недостатком. Поэтому для повышения точности вычислений часто производят округление результата умножения, вследствие чего погрешность становится знакопеременной.

Для округления произведения длина сумматора частичных произведений обычно увеличивается на один разряд. После образования произведения к этому дополнительному разряду прибавляется 1. Если дополнительный разряд произведения был равен 0, то произведение в основных разрядах сумматора получается с недостатком. Если дополнительный разряд был равен 1, то в результате переноса 1 из дополнительного разряда к основным разрядам сумматора добавляется единица и произведение получается с избытком, при этом максимальное значение погрешности произведения равно половине 1 младшего разряда.

Отметим, что при любом методе умножения операция обычно начинается с анализа на 0 сомножителей. При равенстве нулю хотя бы одного сомножителя умножение не производится, а произведению присваивается нулевое значение.

Рассмотрим более подробно наиболее распространенный метод умножения целых чисел, начиная с младших разрядов, со сдвигом суммы частичных произведений вправо.

*Алгоритм умножения чисел, представленных в прямом коде, начиная с младших разрядов, со сдвигом суммы частичных произведений вправо:*

1. Берутся модули от сомножителей.
2. Исходное значение суммы частичных произведений принимается равным 0.

3. Если анализируемая цифра множителя равна 1, то к сумме частичных произведений прибавляется множимое; если эта цифра равна 0, прибавление не производится.

4. Производится сдвиг суммы частичных произведений вправо на один разряд.

5. Пункты 3 и 4 последовательно выполняются для всех цифровых разрядов множителя, начиная с младшего.

6. Произведению присваивается знак плюс, если знаки сомножителей одинаковы, в противном случае — знак минус.

Детальное рассмотрение метода умножения начнем со случая умножения целых положительных чисел, а затем перейдем к числам со знаками.

Пусть  $X$  — множимое и  $Y$  — множитель, тогда, так как числа имеют  $n-1$  цифровых разрядов с весами, изменяющимися от  $2^0$  для разряда с номером  $n-1$  до  $2^{n-2}$  для разряда с номером 1, произведение  $Z=XY$  можно представить в виде

$$\begin{aligned} Z &= XY = X(y_1 2^{n-2} + y_2 2^{n-3} + \dots + y_{n-2} 2^1 + y_{n-1} 2^0) = \\ &= 2^{n-1} X \{y_1 2^{-1} + y_2 2^{-2} + \dots + y_{n-2} 2^{-(n-2)} + y_{n-1} 2^{-(n-1)}\} = \\ &= 2^{n-1} \{ \dots ((0 + X y_{n-1}) 2^{-1} + X y_{n-2}) 2^{-1} + \dots + X y_1) 2^{-1} \} = \\ &= 2^{n-1} A, \end{aligned} \tag{7.1}$$

где  $y_i = (0, 1)$  — значение  $i$ -го разряда множителя  $Y$ .

Для получения произведения  $Z$  производятся вычисления по приведенной формуле, при этом  $X y_i$  равно либо 0 (при  $y_i = 0$ ), либо  $X$  (при  $y_i = 1$ ), а умножение на  $2^{-1}$  осуществляется путем сдвига числа на один разряд вправо.

Вычислив  $A$  и перенеся условную запятую на  $n-1$  разряд вправо (умножив  $A$  на  $2^{n-1}$ ), получим  $Z$ .

Особенностью умножения целых чисел является необходимость представления результата умножения двух  $n$ -разрядных слов двойным словом, при этом число цифровых разрядов двойного слова  $2N-1$  на 1 больше числа  $2n-2$  цифровых разрядов

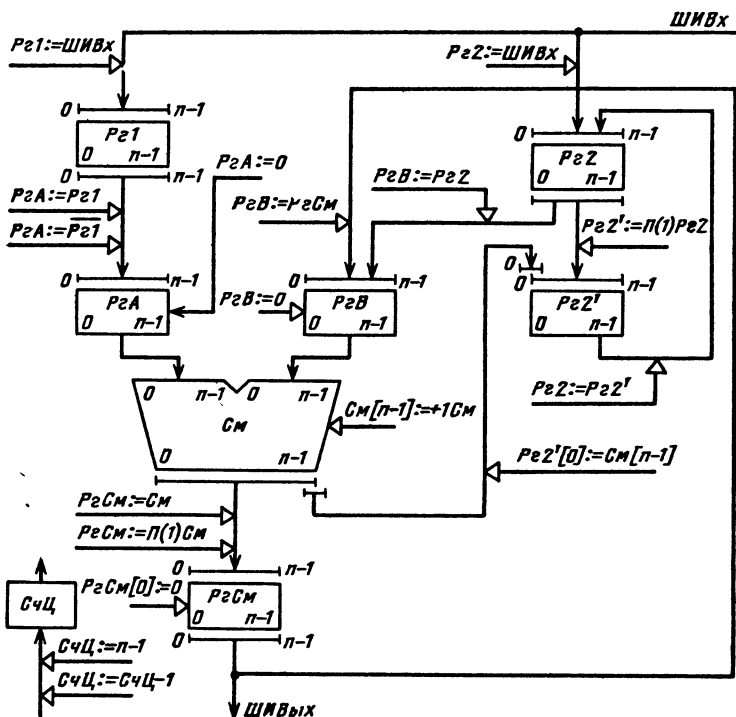


Рис. 7.4. Структурная схема АЛУ для умножения целых чисел

произведения двух чисел, содержащих  $n-1$  цифровых разрядов. В связи с этим после получения результата в формате двойного слова необходимо дополнительно сдвинуть его цифровые разряды на один разряд вправо, чтобы правильно расположить произведение в разрядной сетке.

На рис. 7.4 представлена структурная схема АЛУ для умножения  $n$ -разрядных целых двоичных чисел. В состав АЛУ входят входной регистр множимого  $Pz1$ , регистры множителя  $Pz2$  и  $Pz2'$ , на которых с помощью косой передачи вправо  $Pz2' := \Pi(1)Pz2$  и передачи  $Pz2 := Pz2'$  выполняется сдвиг множителя вправо; сумматор  $См$  для образования суммы частичных произведений, входные и выходной регистры сумматора соответственно  $PzA$ ,  $PzB$  и  $PzCm$ , на которых соответственно хранится текущее значение и образуется новое значение суммы, счетчик циклов  $C4Ц$ .

Работа АЛУ при умножении целых положительных чисел описывается микропрограммой, приведенной на рис. 7.5. Перво-

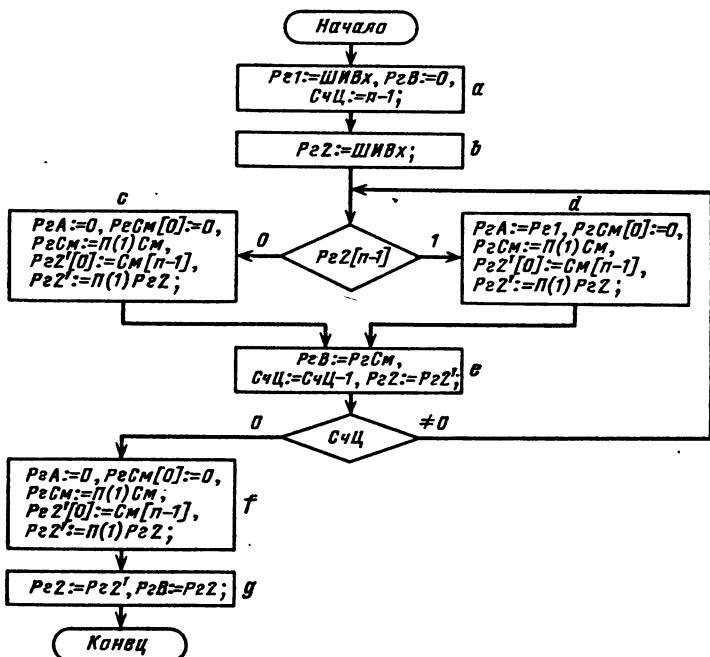


Рис. 7.5. Микропрограмма умножения целых положительных чисел

начально на  $Pz1$  поступает множимое, регистр  $PzB$ , хранящий сумму частичных произведений, обнуляется, а в счетчик циклов заносится число обрабатываемых цифровых разрядов (блок  $a$  микропрограммы). Затем на  $Pz2$  поступает множитель (блок  $b$ ). На этом завершается процедура начальных установок, и начинается процесс вычисления сумм частичных произведений.

В зависимости от значения 0 или 1 младшего разряда множителя к частичному произведению прибавляется либо 0, либо  $X$ , для чего соответствующее значение присваивается регистру  $PzA$ ; полученная сумма умножается на  $2^{-1}$  путем передачи кода с выхода сумматора на  $PzСМ$  со сдвигом на один разряд вправо.

Одновременно множитель подготавливается к перемещению в  $Pz2$  так, чтобы на месте анализируемого младшего разряда  $Pz2$  оказался следующий разряд множителя. Для этого содержимое  $Pz2$  переносится в  $Pz2'$  со смещением вправо на один разряд. Разряд 0  $Pz2'$  при этом остается свободным, и в него заносится младший разряд суммы, выходящий при сдвиге за пределы  $PzСМ$  ( $Pz2'[0] := СМ[n-1]$ ). В следующем такте (блок  $e$ ) завершается сдвиг множителя путем занесения содержимого  $Pz2'$  в  $Pz2$ , и в  $PzB$  образуется сдвинутая сумма частичных

произведений. Кроме того, в этом такте уменьшается на 1 содержимое счетчика циклов.

Когда счетчик циклов установится в 0, в  $P_2Cm$  и  $P_22$  будут храниться соответственно старшие и младшие разряды произведения, требующие сдвига на один разряд вправо для правильного расположения в формате двойного слова. После выполнения этого сдвига (блоки  $f$  и  $g$ ) результат операции из  $P_2Cm$  и  $P_2V$  поступает на  $ШИВых$ . Микрооперации выдачи произведения на  $ШИВых$  на рисунке не показаны.

Умножение чисел со знаком можно свести к взятию модулей от сомножителей, их перемножению (см. микропрограмму на рис. 7.5) и формированию знакового разряда произведения.

При использовании для представления чисел дополнительно кода умножения выполняется обычно непосредственно над дополнительными кодами сомножителей.

Покажем, каким образом это может быть осуществлено.

Сначала рассмотрим случай  $X \leq 0$  и  $Y > 0$ .

Формула (7.1) не меняется, но для получения отрицательно-го  $Z$  в дополнительном коде следует при  $X < 0$  подсуммирование  $X$  производить в дополнительном коде. Это естественным образом происходит при представлении отрицательных чисел в дополнительном коде и не требует специальных схемных решений. Кроме того, следует учесть, что при умножении на  $2^{-1}$  происходят обычный сдвиг положительных чисел и модифицированный сдвиг (с занесением 1 в старший разряд сдвигаемого числа) отрицательных чисел. В итоге  $Z \geq 0$  будет представлено прямым кодом, а  $Z < 0$  — дополнительным.

Рассмотрим случай  $X \leq 0$  и  $Y < 0$ . Если  $Y < 0$ , то  $Y_{\text{доп}} = 2^n - |Y|$ . Тогда в цифровых разрядах кода  $Y$  будет представлено число  $2^{n-1} - |Y|$ , так как вес знакового разряда составляет  $2^{n-1}$ . Если умножить  $X$  на цифровые разряды  $Y_{\text{доп}}$  так, как умножали при  $Y > 0$ , то получим

$$Z' = X (2^{n-1} - |Y|) = -|Y|X + X \cdot 2^{n-1}.$$

Псевдопроизведение  $Z'$  больше истинного на  $X \cdot 2^{n-1}$ . Поэтому можно получать произведение, вычисляя псевдопроизведение и вычитая из него  $X \cdot 2^{n-1}$ .

Поправка вносится при  $C4C=0$  перед последним сдвигом, ставящим результат на свое место в двойном слове. Заметим, что после внесения этой поправки не возникает переполнения, а следовательно,  $P_2Cm[0]$  указывает знак результата, определяющий, какой производится сдвиг (обычный или модифицированный).

*Алгоритм умножения, начиная с младших разрядов множителя, со сдвигом суммы частичных произведений и использованием дополнительного кода для представления чисел:*

1. Исходное значение суммы частичных произведений принимается равным 0.

2. Если анализируемая цифра множителя равна 1, то к сумме частичных произведений прибавляется <sup>1</sup> множимое в том коде, в котором оно представлено; если эта цифра равна 0, прибавление не производится.

3. Сумма частичных произведений сдвигается на один разряд вправо, при этом, если сумма отрицательна, осуществляется модифицированный сдвиг.

4. Пункты 2 и 3 последовательно выполняются для всех цифровых разрядов множителя, начиная с младшего.

5. Если множитель — положительное число, полученный результат представляет собой произведение. Если множитель отрицателен, то для получения произведения к результату прибавляется <sup>1</sup> множимое с обратным знаком.

Если результат размещается в двойном слове, то он предварительно сдвигается на один разряд вправо.

Произведение получается в дополнительном коде.

Для выполнения такого умножения можно использовать АЛУ, представленное на рис. 7.4. Микропрограмма умножения приведена в [22а].

## **7.4. Методы ускорения умножения**

Методы ускорения умножения делятся на аппаратурные и логические. Как те, так и другие требуют дополнительных затрат оборудования. При использовании аппаратурных методов дополнительные затраты оборудования прямо пропорциональны числу разрядов в операндах. Эти методы вызывают усложнение схемы АЛУ.

Дополнительные затраты оборудования при реализации логических методов ускорения умножения не зависят от разрядности операндов. Усложняется в основном схема управления АЛУ. В ЭВМ для ускорения умножения часто используются комбинации этих методов.

К аппаратурным методам ускорения умножения относятся ускорение выполнения операций сложения и сдвига, введение дополнительных цепей сдвига, позволяющих за один такт про-

---

<sup>1</sup> Прибавление производится с выравниванием складываемых чисел по старшим разрядам.

изводить сдвиг информации в регистрах сразу на несколько разрядов, совмещение по времени операций сложения и сдвига, построение комбинационных схем множительных устройств, реализующих «табличное» умножение.

Среди логических методов наиболее распространены в настоящее время методы, позволяющие за один шаг умножения обработать несколько разрядов множителя.

Рассмотрим один из способов умножения на два разряда множителя, начиная с его младших разрядов. В зависимости от результата анализа пары разрядов множителя предусматриваются следующие действия. При 00 производится простой сдвиг на два разряда вправо суммы частичных произведений. При 01 к сумме частичных произведений прибавляется одинарное множимое и сумма частичных произведений сдвигается на два разряда вправо. При 10 прибавляется удвоенное множимое и сумма частичных произведений сдвигается на два разряда вправо. При 11 из суммы частичных произведений вычитается одинарное множимое и сумма частичных произведений сдвигается на два разряда вправо. Тогда в первых трех случаях результат получается правильный, а в последнем — неправильный, он должен быть скорректирован на следующем шаге.

Поскольку при 11 из суммы частичных произведений вычитается одинарное множимое вместо прибавления утроенного множимого, для корректировки результата к сумме частичных произведений перед выполнением сдвига надо было бы прибавить учетверенное множимое. Но после сдвига на два разряда вправо сумма частичных произведений уменьшается в 4 раза, так что для корректировки его на следующем шаге должно быть прибавлено одинарное множимое.

Это учитывается при обработке следующей пары разрядов. Если следующая пара 00, то она обрабатывается как 01, если 01, то как 10, если 10, то как 11, если 11, то как 00 и фиксируется необходимостью коррекции при обработке следующей пары. Удвоенное множимое может быть получено его сдвигом. Признак необходимости коррекции может запоминаться в отдельном триггере коррекции.

Правила обработки пар разрядов множителя с учетом признака коррекции сведены в табл. 7.1.

После обработки каждой комбинации содержимое регистра множителя и сумматора частичных произведений сдвигается на два разряда вправо.

Данный метод умножения требует корректировки результата, если старшая пара разрядов множителя 11 или 10 и состояние триггера коррекции являются единичными. В этом случае к полученному произведению должно быть добавлено множимое.

Таблица 7.1

Пара разрядов множителя	Признак коррекции из предыдущей пары	Признак коррекции для следующей пары	Знак действия	Кратность множителю
00	0	0	—	0
01	0	0	+	1
10	0	0	+	2
11	0	1	—	1
00	1	0	+	1
01	1	0	+	2
10	1	1	—	1
11	1	1	—	0

Аналогичным образом можно организовать умножение с обработкой за шаг большего числа разрядов множителя.

Если, например, умножение выполняется с обработкой за шаг четырех разрядов множителя, то на каждом шаге умножения анализируется очередная тетрада множителя и для каждой пары разрядов тетрады может определяться частичное произведение. Полученные два частичных произведения одновременно суммируются со сдвинутой на четыре разряда вправо текущей суммой частичных произведений.

Более детально с методами логического ускорения умножения можно ознакомиться в [22а].

Все более широкое распространение, особенно в микропроцессорных системах, получают в настоящее время аппаратные методы ускорения умножения, основанные на использовании комбинационных схем множительных устройств. Такие схемы реализуются в виде отдельных БИС или их композиции. Так, в составе БИС серии КР1802 имеется БИС-умножитель двух 16-разрядных операндов, вырабатывающий за один такт произведение двойной длины в прямом или дополнительном коде.

## 7.5. Структура АЛУ для деления чисел с фиксированной точкой

Деление в ЭВМ обычно сводится к выполнению последовательности вычитаний делителя сначала из делимого, а затем из образующихся в процессе деления частичных остатков и сдвига частичных остатков.

Алгоритмы деления аналогичны алгоритму деления при ручном счете. Рассмотрим особенности деления на примере деления целых чисел. Пусть  $Z = X/Y$ , где  $X$  — делимое, представляемое



обычно двойным словом ( $2n-1$  цифровых разрядов),  $Y$  — делитель и  $Z$  — частное, представляемые словами, содержащими  $n-1$  цифровых разрядов.

Будем для простоты считать, что делению подвергаются целые числа, представляемые в прямом коде. Так как  $Z$  — слово, то должно выполняться неравенство  $|Z| < 2^{n-1}$ . Это возможно при  $(|X'| - |Y|) < 0$ , где  $|X'| = |X| \cdot 2^{-(n-1)}$ . Для получения  $|X'| - |Y|$  следует вычесть из делимого  $|X|$  делитель  $Y$ , выровняв их так, чтобы младший разряд  $|Y|$  был под  $n$ -м разрядом. Этого можно добиться, сдвинув  $|Y|$  относительно  $|X|$  на  $n-1$  разряд влево.

Если результат пробного вычитания больше 0, то  $|Z| \geq 2^{n-1}$  и деление невозможно, если он меньше 0, то можно выполнить деление.

Рассмотрим пример деления. Пусть делимое  $X$  и делитель  $Y$  в прямом коде есть  $X_{\text{пр}} = 10101001$  и  $Y_{\text{пр}} = 0111$ , тогда  $|X| = 00101001$  и  $|Y| = 0111$ . Частное  $Z$  должно быть представлено прямым кодом с четырьмя двоичными разрядами, старший из которых представляет собой знак и в  $|Z|$  должен быть равен 0. Выполним деление  $|X|$  на  $|Y|$  обычным способом:

$$\begin{array}{r} \text{Пробное вычитание} \left\{ \begin{array}{r} \begin{array}{r} 00101001 \\ - 0111 \\ \hline (< 0) \\ 001010 \\ - 0111 \\ \hline 00110 \end{array} > 0 \\ \begin{array}{r} 00110 \\ - 0111 \\ \hline (< 0) \\ 001101 \\ - 0111 \\ \hline 0110 \end{array} > 0 \end{array} \right. \begin{array}{r} 0111 \\ 0101 \\ \hline \end{array} \\ \underbrace{0110}_{\text{Остаток}} \end{array}$$

В соответствии с правилами деления очередной цифрой частного является 1, если после вычитания делителя из остатка получается положительный результат, и 0, если отрицательный. В последнем случае восстанавливается остаток, который был до вычитания. Затем делитель смещается на разряд вправо, и процедура повторяется.

В рассматриваемом примере по отрицательному результату пробного вычитания согласно общему правилу фиксируется

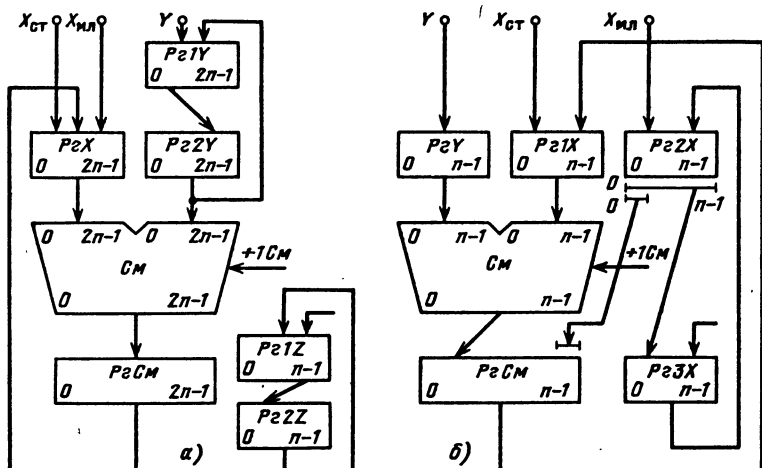


Рис. 7.6. Методы выполнения деления:

а — со сдвигаемым делителем; б — с неподвижным делителем

цифра частного  $z_0 = 0$  для единообразия процедуры деления. На ее место после завершения деления модулей заносится знак результата.

Так как знаки делимого  $X$  и делителя  $Y$  в примере различны, то знак  $Z$  отрицателен, поэтому от  $|Z| = 0101$  переходим к прямому коду  $Z_{пр} = 1101$ .

Реализовать деление можно двумя основными способами.

1. Деление с неподвижным делимым и сдвигаемым вправо делителем. Этот способ деления основан на прямом копировании действий при ручном делении. Структура АЛУ для деления имеет вид, представленный на рис. 7.6, а. Сначала делимое  $X$  заносится в  $PzX$ , а делитель  $Y$  — в старшие разряды  $Pz1Y$ . Делитель сдвигается вправо путем косой передачи из  $Pz1Y$  в  $Pz2Y$  и прямой передачи из  $Pz2Y$  в  $Pz1Y$ . Вычитание делителя выполняется подсуммированием дополнительного кода отрицательного делителя. Цифры частного, определяемые по знаку частичных остатков, фиксируются в регистре  $Pz1Z$  путем последовательного занесения их в младший разряд  $Pz1Z$  и сдвига содержимого  $Pz1Z$  с помощью косой передачи в  $Pz2Z$  и прямой из  $Pz2Z$  в  $Pz1Z$ .

Недостатком такого АЛУ является двойная длина сумматора и его регистров.

2. Деление с неподвижным делителем и сдвигаемым влево делимым. Этот способ позволяет строить АЛУ с сумматором одинарной длины (рис. 7.6, б). Здесь неподвижный делитель  $Y$  хранится в  $PzY$ , а делимое  $X$ , сдвигаемое влево относи-

тельно  $Y$ , находится в двух регистрах: старшие разряды  $X$  — в  $P_21X$ , а младшие — в  $P_22X$ . Деление начинается со сдвига влево делимого  $X$  путем косой передачи его в  $P_2Cm$  и  $P_23X$  и соответствующих прямых передач в  $P_21X$  и  $P_22X$ . Далее на вход сумматора подается сдвинутое влево делимое, образуется частичный остаток путем подсуммирования дополнительного кода отрицательного делителя, и очередная цифра частного заносится в освободившийся при сдвиге  $X$  разряд  $P_22X$ .

Арифметическо-логическое устройство рассмотренного типа широко используется для деления.

*Алгоритм деления с неподвижным делителем с восстановлением остатка:*

1. Берутся модули от делимого и делителя.
2. Исходное значение частичного остатка полагается равным старшим разрядам делимого.
3. Частичный остаток удваивается путем сдвига на один разряд влево, при этом в освобождающийся при сдвиге младший разряд частичного остатка заносится очередная цифра делимого.
4. Из сдвинутого частичного остатка вычитается делитель и анализируется знак результата вычитания.
5. Очередная цифра модуля частного равна 1, если результат вычитания положителен, и 0, если отрицателен. В последнем случае значение остатка восстанавливается до того, которое было до вычитания.
6. Пункты 3—5 последовательно выполняются для получения всех цифр модуля частного.
7. Знак частного плюс, если знаки делимого и делителя одинаковы, и минус в противном случае.

Рассмотренный метод деления носит название *деления с восстановлением остатка*. Недостатком этого метода является необходимость введения специального такта для восстановления остатка.

Обычно в ЭВМ для деления используется другой метод — *деление без восстановления остатка*.

*Алгоритм деления с неподвижным делителем без восстановления остатка:*

Пункты 1—3 совпадают с аналогичными пунктами алгоритма деления с восстановлением остатка.

4. Из сдвинутого частичного остатка вычитается делитель, если остаток положителен, и к сдвинутому частичному остатку прибавляется делитель, если остаток отрицателен.

5. Очередная цифра модуля частного равна 1, если результат положителен, и 0, если отрицателен.

Пункты 6, 7 совпадают с аналогичными пунктами предыдущего алгоритма.

Можно показать, что частичные остатки после выполнения сложения при делении без восстановления остатка получаются такими же, как и после сдвига восстановленного остатка при делении с восстановлением остатка.

Действительно, поскольку сдвиг частичного остатка на один разряд влево эквивалентен умножению его на два, то получим

$$2a - b = 2(a - b) + b, \quad (7.2)$$

где  $a$  — частичный остаток;  $b$  — делитель.

Деление без восстановления остатка всегда требует для получения одной цифры частного только сложения или вычитания и сдвига частичного остатка. После завершения всех циклов деления выдается результат, при этом если остаток отрицателен, то он восстанавливается путем подсуммирования  $Y$ .

Деление чисел, представленных дополнительными кодами, можно осуществлять, не переходя к модулям, при этом алгоритм деления оказывается подобным рассмотренным.

Отличия заключаются в следующем (для деления без восстановления остатка):

1. Так как делимое и делитель могут иметь разные знаки, то действия с частичным остатком (прибавление или вычитание  $Y$ ) зависят от знаков остатка и делителя и определяются в соответствии с табл. 7.2.

Если знак остатка совпадает со знаком делителя, то  $z_i = 1$ , иначе  $z_i = 0$ .

2. Если  $X > 0$  и  $Y < 0$ , частное необходимо увеличить на 1.

Если  $X < 0$  и  $Y > 0$ , частное необходимо увеличить на 1 при остатке от деления, не равном 0.

Если  $X < 0$  и  $Y < 0$ , частное необходимо увеличить на 1 при остатке от деления, равном 0.

Деление правильных дробей выполняется так же, как и деление целых чисел. Разница заключается только в том, что делимое имеет, как правило, такую же длину, как и делитель. Однако можно предположить, что делимое имеет еще  $n$  младших разрядов, равных 0. Тогда становится ясно, что алгоритм деления дробей ничем не отличается от алгоритма деления целых чисел.

Микропрограммы для операции деления приведены в [22a].

Т а б л и ц а 7.2

Знак остатка	Знак делителя	Действие
+	+	Вычитание $Y$
+	—	Прибавление $Y$
—	+	Прибавление $Y$
—	—	Вычитание $Y$

## 7.6. Устройства для выполнения логических операций

В состав операций, реализуемых ЭВМ, входят следующие поразрядные логические операции: суммирование по модулю 2, логическое умножение И, логическое сложение ИЛИ.

В результате поразрядной логической операции над словами  $X$  и  $Y$  формируется слово  $Z$ , в котором  $Z[i] = X[i] * Y[i]$ , где  $*$  — символ выполняемой операции.

Для выполнения логических операций можно воспользоваться устройством, структурная схема которого приведена на рис. 7.7. Исходные операнды размещаются в регистрах  $Pz1$  и  $Pz3$ , откуда побайтно можно переносить их содержимое в  $PzC$  и  $PzD$  соответственно. Для реализации требуемых логических операций используется схема однобайтовых логических операций СОЛО, входящая в состав АЛУ и являющаяся комбинационной схемой, позволяющей реализовать поразрядные операции логического умножения И, логического сложения ИЛИ и суммирования по модулю 2 над двумя однобайтовыми операндами. Результат обработки байт фиксируется на байтовом регистре  $PzCOLO$ , из которого результат можно отправлять в выходной регистр  $PzCm$ <sup>1</sup>.

Помимо указанных логических операций в СОЛО выполняется операция сравнения двух байт, используемая, в частности, при обработке порядков в операциях сложения и вычитания над числами с плавающей запятой.

Восьмиразрядная схема однобайтовых логических операций (рис. 7.8) состоит из восьми схем поразрядной обработки СПО и схемы сравнения слов длиной 1 байт.

На выходах  $COLO \cdot K_0 - K_7$  формируется поразрядная конъюнкция двух байт, поданных на схему. На выходах  $M_0 - M_7$  образуется поразрядная сумма по модулю 2.

На  $Вых1$  и  $Вых2$  формируются сигналы, определяющие результат сравнения байт по численному значению в соответствии со следующим правилом:

$Вых1$	$Вых2$	Результат сравнения
1	1	$D < C$
0	1	$D > C$
0	0	$D = C$

---

<sup>1</sup> Структурные схемы, рассматриваемые в гл. 7, являются фрагментами общей структурной схемы АЛУ, обеспечивающего выполнение всей совокупности операций процессора.

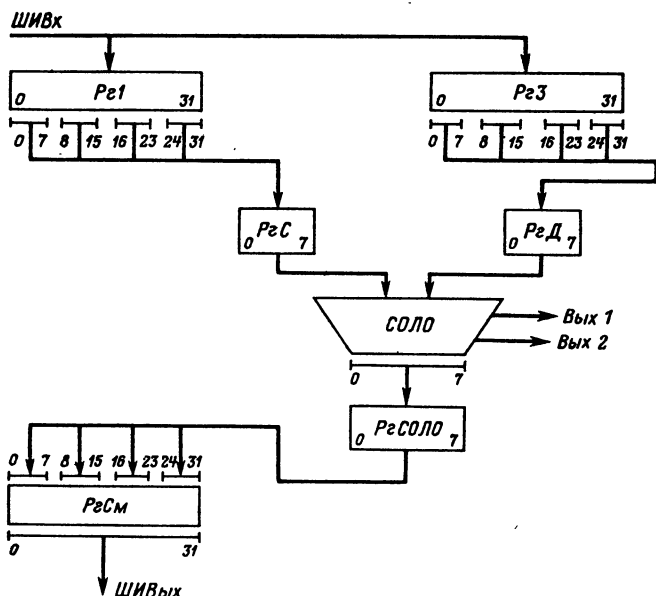
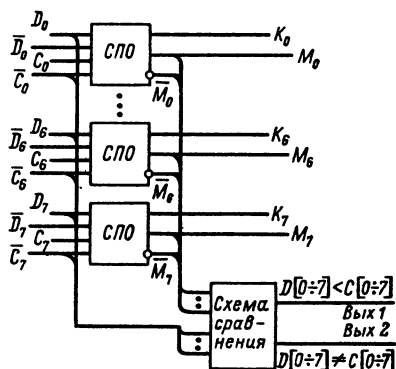


Рис. 7.7. Устройство для выполнения логических операций

Рис. 7.8. Комбинационная схема поразрядной обработки



На выходе  $K_i$  образуется конъюнкция

$$K_i = \overline{C_i} \vee \overline{D_i} = C_i D_i,$$

где  $C_i$  и  $D_i$  — значения  $i$ -х разрядов кодов, хранящихся соответственно на регистрах  $PzC$  и  $PzD$ . На выходе  $M_i$  формируется значение суммы по модулю 2 в соответствии с выражением

$$M_i = \overline{C_i D_i} \vee C_i D_i = C_i \oplus D_i.$$

Выходы  $K_i$  и  $M_i$  соединены со входами соответствующих разрядов регистра  $P_2\text{СОЛО}$ . При выполнении микрооперации  $P_2\text{СОЛО} := P_2C \wedge P_2D$  в  $P_2\text{СОЛО}$  заносятся значения состояний выходов  $K_i$ . При выполнении микрооперации  $P_2\text{СОЛО} := P_2C \oplus P_2D$  в  $P_2\text{СОЛО}$  заносятся значения состояний выходов  $M_i$ . Если одновременно выполняются две микрооперации  $P_2\text{СОЛО} := P_2C \wedge P_2D$  и  $P_2\text{СОЛО} := P_2C \oplus P_2D$ , в  $P_2\text{СОЛО}$  заносится результат поразрядной операции ИЛИ  $P_2C \vee P_2D$ .

## 7.7. Особенности операций десятичной арифметики

Арифметические операции над десятичными числами (сложение, вычитание, умножение, деление) выполняются аналогично операциям над целыми двоичными числами. Основой АЛУ десятичной арифметики является сумматор двоично-десятичных кодов. Такой сумматор, как правило, строится на основе двоичного путем добавления некоторых цепей.

Рассмотрим, каким образом можно выполнить сложение двоично-десятичных кодов. Пусть необходимо сложить модули двух двоично-десятичных чисел  $X$  и  $Y$ . Первое слагаемое  $X$  преобразуем в код с избытком 6 (обозначим  $X_6$ ), получаемый путем прибавления к каждой цифре  $X$  двоичного числа 6. Переход от  $X$  к  $X_6$  изменяет все тетрады  $X$  так, что в каждой тетраде  $X_6$  находится число 6—15.

Складывая  $X_6$  и  $Y$  по правилам двоичного сложения, получаем результат  $Z'$ . В  $Z'$  одни тетрады совпадают, а другие не совпадают с тетрадами двоично-десятичной суммы  $Z$ .

Если результат сложения в  $i$ -м разряде  $X[i] + Y[i] + P[i] \geq 10$ , где  $P[i]$  — десятичный перенос в  $i$ -й разряд, то  $i$ -я десятичная цифра  $Z[i] = X[i] + Y[i] + P[i] - 10$  и  $P[i+1] = 1$ , где  $P[i+1]$  — десятичный перенос в  $(i+1)$ -й разряд.

Для  $Z'[i]$  в этом случае получаем

$$Z'[i] = X_6[i] + Y[i] + P[i] - 16 = 6 + X[i] + Y[i] + P[i] - 16 = Z[i].$$

$$\underbrace{\qquad\qquad\qquad}_{\geq 10}$$

$$\qquad\qquad\qquad \geq 16$$

При этом возникает перенос в  $(i+1)$ -ю тетраду.

Если  $i$ -я десятичная цифра  $Z(i)$  должна получаться из

$X[i] + Y[i] + P[i] < 10$ , то  $Z[i] = X[i] + Y[i] + P[i]$  и  $P[i+1] = 0$ .

Для  $Z'(i)$  в этом случае получаем

$$Z'[i] = X6[i] + Y[i] + P[i] = 6 + X[i] + Y[i] + P[i] = Z[i] + 6.$$

$$\begin{array}{c} \hline < 10 \\ \hline < 16 \end{array}$$

Перенос в  $(i+1)$ -ю тетраду здесь не возникает ( $P[i+1] = 0$ ), так как  $Z'[i] < 16$ .

Таким образом, складывая  $X6$  и  $Y$  как двоичные числа, получаем  $Z'$ . В  $Z'$  тетрады, из которых возникал перенос, совпадают с тетрадами двоично-десятичного результата  $Z$ , а тетрады, из которых не было переноса при сложении, представлены с избытком 6. Для получения суммы  $Z$  необходимо откорректировать  $Z'$  путем уменьшения на 6 тех тетрад  $Z'$ , из которых не было переноса при сложении  $X6$  и  $Y$ .

Вычитание 6 из тетрад, требующих коррекции, можно реализовать путем подсуммирования 10 с одновременным игнорированием переноса, возникающего при этом из тетрад. Если  $Z'[i]$  нуждается в коррекции, то  $Z'[i] = Z[i] + 6$ . Поэтому  $Z'[i] + 10 \geq 16$ , значит, после прибавления 10 из тетрады возникнет перенос, т. е. в тетраде останется  $(Z'[i] + 10) - 16 = Z'[i] - 6$ .

Вычитание двоично-десятичных модулей  $X - Y$  осуществляется следующим образом.

Все разряды  $Y$  инвертируются, что дает дополнение каждой цифры  $Y$  до 15, при этом получается обратный код двоично-десятичного ( $-Y$ ) с избытком 6, обозначенный  $Y_{\text{обр}}6$ . Затем, складывая  $X + Y_{\text{обр}}6$  и прибавляя 1 к младшему разряду, получаем  $Z'$ . Результат  $Z'$  является положительным числом, если из старшей тетрады его возникает перенос, при этом  $Z'$  корректируется по тем же правилам, что и при сложении модулей.

Если из старшей тетрады  $Z'$  нет переноса, то получен отрицательный результат, представленный в дополнительном коде. В этом случае код  $Z'$  инвертируется и к нему прибавляется 1 младшего разряда. Новое  $Z'$  корректируется, при этом к тетрадам, из которых возникал перенос при получении  $(X + Y_{\text{обр}}6 + 1)$ , прибавляется 10, а к остальным не прибавляется.

Выполнение сложения и вычитания чисел со знаками сводится к выполнению сложения или вычитания модулей путем определения фактической выполняемой операции по знакам операндов и виду выполняемой операции. Знак результата определяется отдельно. Например, при  $X < 0$  и  $Y < 0$  вычитание  $X - Y$  заменяется вычитанием  $|Y| - |X|$ . Затем знак результата меняется на противоположный знаку  $(|Y| - |X|)$ .



Двоично-десятичное умножение сводится к образованию и многократному сложению частичных двоично-десятичных произведений. Умножение двоично-десятичных чисел выполняется следующим образом:

- 1) сумма частичных произведений полагается равной нулю;
- 2) анализируется очередная цифра (тетрада) множителя, и множимое прибавляется к сумме частичных произведений столько раз, какова цифра множителя;
- 3) сумма частичных произведений сдвигается вправо на 1 тетраду, и повторяются действия, указанные в п. 2, пока все цифры множителя не будут обработаны.

Для ускорения умножения часто отдельно формируются кратные множимого  $8X$ ,  $4X$ ,  $2X$  и  $1X$ , при наличии которых уменьшается число сложений при выполнении п. 2.

Двоично-десятичное деление выполняется путем многократных вычитаний, подобно тому как это делается при обычном делении.

Двоично-десятичные АЛУ часто строятся как последовательно-параллельные, осуществляющие последовательную обработку байт.

## 7.8. Операции над числами с плавающей точкой

Арифметические операции над числами с плавающей точкой более сложны, чем операции над числами с фиксированной точкой.

Сложение (вычитание) чисел с плавающей точкой (в предположении, что  $|X| \geq |Y|$ ) выполняется согласно выражению

$$\begin{aligned} Z = X \pm Y &= S^{p_x} q_x \pm S^{p_y} q_y = \\ &= S^{p_x} \left( q_x \pm \frac{q_y}{S^{(p_x - p_y)}} \right) = S^{p_z} q_z. \end{aligned} \quad (7.3)$$

*Алгоритм сложения и вычитания чисел с плавающей точкой:*

1. Производится выравнивание порядков чисел. Порядок меньшего (по модулю) числа принимается равным порядку большего, а мантисса меньшего числа сдвигается вправо на число  $S$ -ичных разрядов, равное разности порядков чисел.

2. Производится сложение (вычитание) мантисс, в результате чего получается мантисса суммы (разности).

3. Порядок результата принимается равным порядку большего числа.

4. Полученная сумма (разность) нормализуется.

**Выравнивание порядков** начинается с их сравнения. Мантисса числа с меньшим порядком при выравнивании сдвигается вправо на число разрядов, равное разности порядков. Если рассматриваемые числа с плавающей точкой имеют  $S=16$ , сдвиг осуществляется шестнадцатиричными разрядами, т. е. каждый сдвиг производится на четыре двоичных разряда.

При сравнении порядков возможны пять случаев:

1)  $p_x - p_y > m$  ( $m$  — число разрядов мантиссы). В качестве результата суммирования сразу может быть взято первое слагаемое, так как при выравнивании порядков все разряды мантиссы второго слагаемого принимают нулевое значение;

2)  $p_y - p_x > m$ . В качестве результата суммирования может быть взято второе слагаемое;

3)  $p_x - p_y = 0$ . Можно приступить к суммированию мантисс;

4)  $p_x - p_y = k_1$  ( $k_1 < m$ ). Мантисса второго слагаемого сдвигается на  $k_1$  разрядов вправо, затем производится суммирование мантисс;

5)  $p_y - p_x = k_2$  ( $k_2 < m$ ). Перед выполнением суммирования мантисс производится сдвиг на  $k_2$  разрядов вправо мантиссы первого слагаемого.

За порядок результата при выполнении суммирования принимается больший из порядков операндов.

**Сложение (вычитание) мантисс** производится по правилам сложения (вычитания) чисел с фиксированной точкой.

**Нормализация суммы (разности)** производится в случае невыполнения условия  $1 > q_z \geq 1/s$ , при этом, если  $q_z \geq 1$ ,  $p_z$  увеличивается на 1, а мантисса  $q_z$  сдвигается на один  $S$ -ичный разряд вправо, что дает  $|q_z| < 1$ . Если  $|q_z| < 1/s$ , то мантисса результата сдвигается на разряд влево при одновременном уменьшении порядка результата на 1. Эти операции производятся до тех пор, пока не станет выполняться условие  $q_z \geq 1/s$ . (При  $q_z = 0$  нормализация не выполняется.)

При получении порядка  $+p_z$ , переполняющего разрядную сетку, должен формироваться сигнал прерывания из-за переполнения порядка. При получении порядка  $-p_z$ , переполняющего разрядную сетку, формируются нулевой результат и признак исчезновения порядка.

В операциях с плавающей точкой в отличие от операций с фиксированной точкой сложение и вычитание выполняются приближенно, так как при выравнивании порядков происходит потеря младших разрядов одного из слагаемых. В этом случае погрешность всегда отрицательна и может доходить до единицы младшего разряда. Чтобы уменьшить погрешность, применяют округление результата. Для этого может быть использован до-

полнительный разряд сумматора, в который после выполнения суммирования добавляется 1.

Умножение чисел с плавающей точкой выполняется в соответствии с формулой

$$Z = XY = S^p q_x S^p q_y = S^{(p_x + p_y)} q_x q_y = S^p q_z. \quad (7.4)$$

При умножении чисел с плавающей точкой порядки сомножителей складываются, а мантиссы перемножаются. Произведение нормализуется, и ему присваивается знак плюс, если сомножители имеют одинаковые знаки, и знак минус, если знаки разные.

Если мантисса множимого или множителя равна 0, то произведению можно присвоить значение 0 без выполнения умножения мантисс. Если при суммировании порядков возникло переполнение и порядок отрицательный, то это означает, что произведение меньше минимального представляемого в машине числа, и в качестве результата операции может быть записан 0 без перемножения мантисс.

Если при суммировании порядков возникает переполнение и порядок положительный, может оказаться, что результат все-таки находится в диапазоне чисел, представляемых в машине, так как при умножении мантисс возможно нарушение нормализации вправо, и после нормализации мантиссы переполнение в порядке может исчезнуть.

Деление чисел с плавающей точкой выполняется в соответствии с формулой

$$Z = S^p q_x / S^p q_y = S^{(p_x - p_y)} q_x / q_y = S^p q_z. \quad (7.5)$$

При делении с плавающей точкой мантисса частного равна частному от деления мантиссы делимого на мантиссу делителя, а порядок частного — разности порядков делимого и делителя. Частное нормализуется, и ему присваивается знак плюс, если делимое и делитель имеют одинаковые знаки, и знак минус, если разные.

Если делимое равно 0, то в частное может быть записан 0 без выполнения деления. Если при вычитании порядков образовалось переполнение с положительным знаком или если делитель равен 0, то деление не производится и формируется сигнал прерывания.

При делении нормализованных чисел с плавающей точкой может оказаться, что мантисса делимого больше мантиссы делителя, и мантисса частного образуется с переполнением. Для устранения этого явления перед делением мантисс нарушают нормализацию делителя сдвигом на разряд влево. Тогда нарушения нормализации частного влево не возникает.

Деление мантисс выполняется, как правило, методом без восстановления остатка аналогично делению целых чисел. Отличие заключается в том, что делимое берется такой же длины, как и делитель. Однако нетрудно заметить, что для дробей можно условно принять, что делимое имеет двойную длину с нулями в разрядах младшей половины числа. После сдвигов влево частичных остатков освобождающиеся разряды всегда заполняются 0 и деление можно выполнять точно так же, как и деление целых чисел.

Более детально алгоритмы и микропрограммы выполнения операций над числами с плавающей точкой изложены в [22а].

## 7.9. Многофункциональное АЛУ

Проектирование АЛУ включает в себя выбор кодов для представления данных, определение алгоритмов выполнения отдельных операций, структур операционных блоков и реализуемых в них наборов микроопераций. Затем производят объединение отдельных операционных блоков и соответствующих наборов микроопераций в один многофункциональный операционный блок или несколько блоков для отдельных групп операций. В многофункциональных АЛУ операции над числами с фиксированной и плавающей точками, десятичными числами и алфавитно-цифровыми полями выполняются в основном одними и теми же схемами, коммутируемыми соответствующим образом. На рис. 7.9 приведена схема многофункционального АЛУ для выполнения совокупности рассматривавшихся арифметических и логических операций. В данной схеме в качестве отдельных фрагментов можно выделить описанные выше АЛУ, при этом для сокращения общего числа связей некоторые связи следует удалить (фактически заменить другими). Так, например, при сложении чисел с фиксированной точкой в отличие от АЛУ на рис. 7.1 в рассматриваемой схеме загрузка  $P_2B$  происходит не от  $ШИВх$ , а от  $P_22$  ввиду того, что связь от  $ШИВх$  к  $P_22$  и далее к  $P_2B$  должна существовать из-за необходимости реализации умножения (см. рис. 7.4). Сумма частичных произведений заносится в  $P_2B$  не непосредственно из  $P_2См$ , а через  $P_23$ , так как загрузка  $P_23$  необходима при выполнении сложения чисел с плавающей точкой и т. п.

Операции двоично-десятичной арифметики в данном АЛУ производятся при помощи двоично-десятичного сумматора  $СмДес$  и побайтной организации обработки.

При выполнении операций над числами с плавающей точкой используются двоичный сумматор  $См$  и схема  $СОЛО$ . При сложении (вычитании) чисел с плавающей точкой первое слагаемое

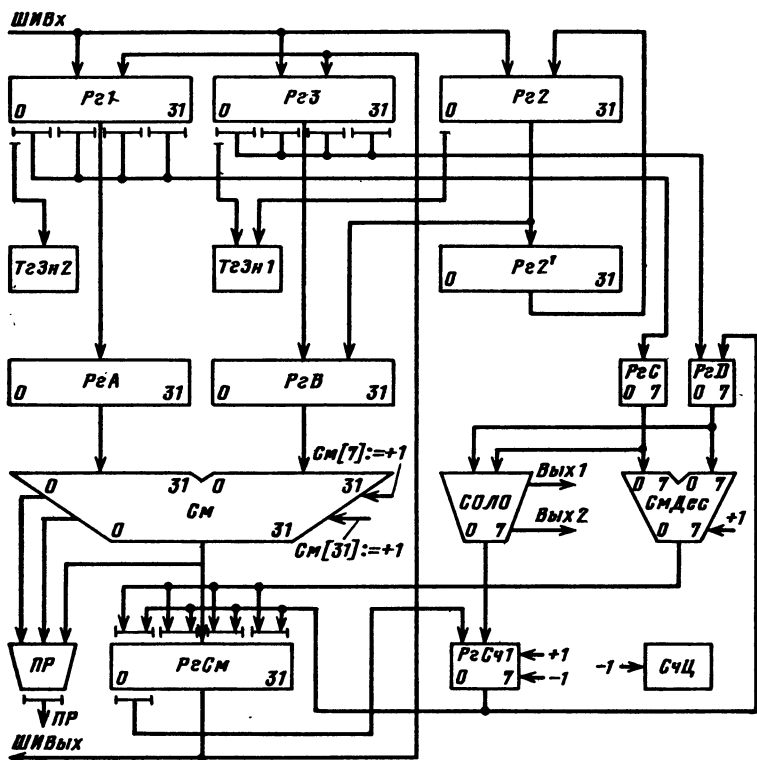


Рис. 7.9. Многофункциональное АЛУ

(уменьшаемое) поступает на входной регистр  $Pz1$ , второе (вычитаемое) — на входной регистр  $Pz3$ . Знаки слагаемых хранятся в триггерах знаков  $Tz3n1$  и  $Tz3n2$ . Смещенные порядки слагаемых пересылаются в регистры  $PzC$  и  $PzD$ . Схема СОЛО применяется для сравнения и выравнивания порядков слагаемых. Сумматор  $См$ , его входные регистры  $PzA$  и  $PzB$  и выходной регистр  $PzCm$  используются при сложении (вычитании) мантисс, а также при передаче мантисс со сдвигом в процедурах выравнивания порядков и нормализации результата.

Выравнивание порядков производится следующим образом. Смещенный порядок числа  $X$  из  $Pz3$  передается в регистр  $PzD$  и в выполняющий роль  $PzCOLO$  счетчик  $PzC41$ , соединенный с выходом СОЛО. Затем в  $PzC$  передается смещенный порядок числа  $Y$ .

После этого начинается сравнение порядков чисел  $X$  и  $Y$  на *СОЛО* и сдвиг мантиссы числа с меньшим порядком вправо, при этом значение смещенного порядка  $Y$  меняется до тех пор, пока он не станет равным смещенному порядку  $X$ . Порядок  $Z$  берется равным большему порядку слагаемых.

Чтобы не делать лишних сдвигов мантиссы, превратившейся в процессе выравнивания порядка в 0, на счетчике циклов *СчЦ* фиксируется предельное число сдвигов, равное числу цифр мантиссы.

При выполнении сдвига на один разряд мантиссы содержимое *СчЦ* уменьшается на 1. При *СчЦ* = 0 сдвиги прекращаются и в качестве результата берется большее слагаемое.

После выравнивания порядков осуществляется сложение мантиссы и (при необходимости) нормализация результата.

При умножении чисел с плавающей точкой используются сумматор *См*, регистр *Рг1* для хранения множимого, регистры *Рг2* и *Рг2'* для приема и сдвига множителя в процессе умножения мантиссы, регистр *РгА*, используемый для передачи на сумматор смещенного порядка множимого при суммировании порядков и для передачи на сумматор мантиссы множимого при умножении мантиссы, регистр *РгВ*, служащий для передачи на сумматор смещенного порядка множителя при суммировании порядков и для хранения текущей суммы частичных произведений при умножении мантиссы, выходной регистр сумматора *РгСм*, фиксирующий результаты суммирований, счетчик *РгСч1*, хранящий смещенный порядок произведения, триггеры знаков сомножителей *ТгЗн1* и *ТгЗн2*.

При выполнении деления чисел с плавающей точкой используются сумматор *См*, регистры *Рг1* и *Рг2* для приема соответственно делителя и делимого, регистры *РгА* и *РгВ* для хранения смещенных порядков делителя и делимого и для хранения мантиссы делителя и частичного остатка при получении мантиссы частного, счетчик *Сч1* для хранения смещенного порядка частного, регистры *Рг2* и *Рг2'* для хранения цифровых разрядов мантиссы частного, триггеры знаков делимого и делителя *ТгЗн1* и *ТгЗн2*. Рассмотренное АЛУ можно считать типичным для ЭВМ общего назначения средней производительности. Несколько иначе строятся АЛУ для малых ЭВМ и микропроцессоров.

## 7.10. Особенности АЛУ микропроцессоров

Для микроЭВМ и микропроцессоров типичной является такая организация, при которой их внутренние регистры используются в различных целях. Система связей у этих регистров, как правило, централизованная (магистральная), обеспечивающая

возможность разнообразных межрегистровых пересылок, в том числе передач в АЛУ и из АЛУ. В связи с этим часто собственные регистры АЛУ (регистры, используемые только для выполнения арифметических и логических операций) в микропроцессорах отсутствуют. Это дает повод рассматривать АЛУ микропроцессоров как комбинационную схему, выполняющую арифметические и логические операции над операндами, находящимися в регистрах микропроцессора. Результат операции засылается в некоторый регистр микропроцессора.

Подобные АЛУ входят в состав большинства микропроцессоров: К580, К1810 и др.

Примером комбинационного АЛУ может служить микросхема 500ИП181, имеющая пять управляющих входов, сигналы которых настраивают ее на выполнение одной из 32 арифметическо-логических одноктактных операций над двумя 4-разрядными операндами.

В процессе выполнения операций комбинационное АЛУ взаимодействует с регистрами микропроцессора, являющимися обычно источниками и приемниками операндов для такого АЛУ, при этом, как правило, один и тот же регистр может рассматриваться и как источник, и как приемник информации. Для реализации такой возможности необходимо осуществлять временное запоминание промежуточных результатов на отдельных регистрах. С этой целью используют либо регистры для кратковременного запоминания операндов, либо регистры для кратковременного запоминания результата.

На рис. 7.10, а показана схема включения комбинационного АЛУ в контур с регистрами микропроцессора для выполнения арифметических операций. В приведенной схеме имеются регистры процессора  $PzП$  (регистр признака результата),  $PzАкк$  (аккумулятор),  $PzI, \dots, Pzm$ , которые могут использоваться произвольным образом, и регистры временного хранения операндов  $PzA$  и  $PzB$ , в которые при выполнении арифметических и логических операций загружаются операнды.

Пусть, например, выполняется операция сложения двух чисел, находящихся в регистрах процессора  $Pzj$  и  $Pzi$ , с засылкой результата в  $Pj$ . Эта операция потребует сначала пересылки содержимого  $Pzj$  и  $Pzi$  в  $PzA$  и  $PzB$ , а затем загрузки результата, сформированного АЛУ, в  $Pzj$ .

Отсутствие  $PzA$  привело бы к возникновению «порочной петли», так как изменения состояний  $Pzj$  влекли бы за собой новые изменения состояний  $Pzj$ .

Для выполнения операции вида  $Pzj := Pzj + Pzj$  возникает необходимость в двух регистрах временного хранения операндов.

Схема на рис. 7.10, б аналогична только что рассмотренной.

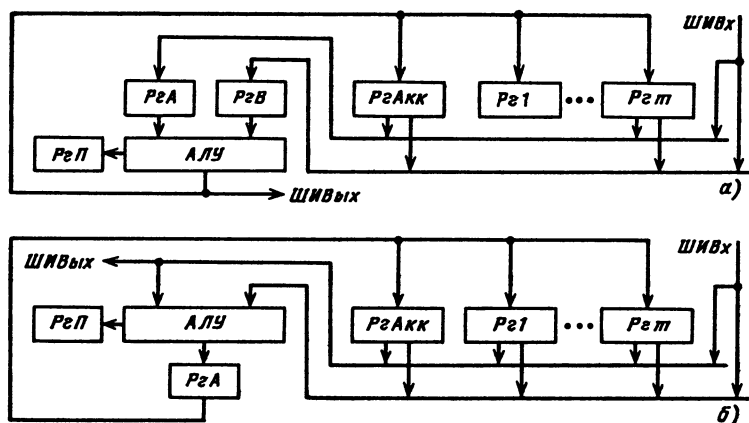


Рис. 7.10. Схемы с комбинационным АЛУ:

а — с регистрами временного запоминания операндов; б — с одним регистром временного запоминания результата

Отличается она лишь тем, что временному хранению подвергается в ней результат операции.

Арифметическо-логические устройства, используемые в рассматриваемых схемах, представляют собой комбинационные схемы, настраиваемые сигналами микроопераций на различные преобразования. Это может быть двоичное или двоично-десятичное сложение, вычитание, логическое умножение и т. п. При написании микропрограмм операций в АЛУ в микрокомандах задаются микрооперации, определяющие выбор источников операндов для АЛУ, настраивающие АЛУ на выполнение различных преобразований и указывающие место занесения результата, сформированного АЛУ.

Рассматриваемые схемы являются фрагментами микропроцессора, в котором те же самые регистры используются и для других целей (см. гл. 9 и 10), что ведет к усложнению микропрограмм.

### Контрольные вопросы

1. Что дает использование дополнительного (обратного) кода для представления чисел при выполнении в АЛУ операции алгебраического сложения?

2. В каких случаях инициируется микрооперация  $+1\text{См}$  (прибавление к сумматору единицы младшего разряда) при выполнении алгебраического сложения?



3. На что указывают следующие комбинации значений переносов из нулевого (знакового) и первого разрядов сумматора:

ПнСм 0	ПнСм 1
0	0
0	1
1	0
1	1

4. Сопоставить методы выполнения умножения двоичных чисел (см. рис. 7.3).

5. Сопоставить методы выполнения деления со сдвигаемым и несдвигаемым делителями.

6. Составить микропрограмму умножения двух  $n$ -разрядных двоичных чисел, начиная с младших разрядов множителя, со сдвигом суммы частичных произведений и использованием дополнительного кода для представления чисел.

7. Из каких этапов (шагов) состоит операция сложения чисел с плавающей точкой? Какой из этих этапов и почему имеет наибольшую продолжительность?

8. Почему использование смещенного кода для представления порядков чисел с плавающей точкой упрощает операцию сложения (вычитания) этих чисел?

9. Какие цепи необходимо добавить к двоичному сумматору для выполнения на нем сложения двоично-десятичных чисел?

## Глава 8

# УПРАВЛЯЮЩИЕ АВТОМАТЫ

## 8.1. Общие сведения

Как указывалось ранее, любое цифровое устройство можно рассматривать состоящим из двух блоков — операционного и управляющего.

Любая команда, операция или процедура, выполняемая в операционном блоке, описывается некоторой микропрограммой и реализуется за несколько тактов, в каждом из которых выполняется одна или несколько микроопераций.

Интервал времени, отводимый на выполнение микрооперации, называется рабочим тактом или просто тактом цифрового устройства. Если все такты имеют одну и ту же длину, то она устанавливается по самой продолжительной микрооперации.

Для реализации команды, операции или процедуры (микропрограммы) необходимо на соответствующие управляющие вхо-

ды операционного блока подать определенным образом распределенную во времени последовательность управляющих функциональных сигналов. Например, для выполнения приведенной в предыдущей главе микропрограммы операции сложения и вычитания необходимо подвести к управляющим входам АЛУ следующую последовательность функциональных сигналов:

Такты

1  $PrP_2B$

2  $PrP_2I$

3  $PrP_2AP$  или  $PrP_2AI$

4  $PrP_2Cm$  или  $PrP_2Cm, +ICm$

5  $PrШИВых$

Подчеркнем, что каждый управляющий функциональный сигнал поступает в начале некоторого такта на соответствующий вход АЛУ, вызывая в этом такте выполнение в АЛУ определенной микрооперации.

Часть цифрового вычислительного устройства, предназначенная для выработки последовательностей управляющих функциональных сигналов, называется *управляющим блоком* или *управляющим устройством*. Генерируемая управляющим блоком последовательность управляющих сигналов задается поступающими на входы блока кодом операции, сигналами из операционного блока, несущими информацию об особенностях операндов, промежуточных и конечного результатов операции, а также синхросигналами, задающими границы тактов.

Формально управляющий блок можно рассматривать как конечный автомат, определяемый:

а) множеством двоичных выходных сигналов

$$V = \{v_1, v_2, \dots, v_m\},$$

соответствующих множеству микроопераций операционного блока. При  $v_i = 1$  возбуждается  $i$ -я микрооперация;

б) множествами входных сигналов  $Z$  и  $U$ :

$$Z = \{z_1, z_2, \dots, z_p\};$$

$$U = \{u_1, u_2, \dots, u_n\},$$

соответствующих задаваемому блоку извне двоичному коду операции ( $Z$ ) и двоичным оповещающим сигналам ( $U$ );

в) множеством подлежащих реализации микропрограмм, устанавливающих в зависимости от значений входных сигналов управляющие сигналы, выдаваемые блоком в определенные такты. По множествам входных и выходных сигналов и микропрограммам определяется множество внутренних состояний блока

$$S = \{Q_0, Q_1, \dots, Q_r\},$$

мощность которого (объем памяти управляющего блока) в процессе проектирования стараются минимизировать.

Сказанное поясняет, почему управляющие блоки называют управляющими автоматами. Поскольку эти автоматы задаются микропрограммами, они часто именуются *микропрограммными автоматами*.

Управляющий автомат может быть задан как автомат Мура:

$$\begin{aligned} Q(t+1) &= A[Q(t), z_1(t), \dots, z_p(t), u_1(t), \dots, u_n(t)]; \\ v_1(t) &= B_1[Q(t)]; \\ &\dots \dots \dots \\ v_m(t) &= B_m[Q(t)]; \end{aligned} \quad (8.1)$$

или как автомат Мили:

$$\begin{aligned} Q(t+1) &= A[Q(t), z_1(t), \dots, z_p(t), u_1(t), \dots, u_n(t)]; \\ v_1(t) &= B_1[Q(t), z_1(t), \dots, z_p(t), u_1(t), \dots, u_n(t)]; \\ v_m(t) &= B_m[Q(t), z_1(t), \dots, z_p(t), u_1(t), \dots, u_n(t)], \end{aligned} \quad (8.1a)$$

где функция переходов  $A$  и функции выходов  $B_i$  определяются заданной микропрограммой.

Существуют два основных типа управляющих автоматов.

1. Управляющий автомат с *жесткой, или схемной, логикой*. Для каждой операции, задаваемой, например, кодом операции команды, строится набор комбинационных схем, которые в нужных тактах возбуждают соответствующие управляющие сигналы. Другими словами, строится конечный автомат, в котором необходимое множество состояний представляется состояниями  $k$  запоминающих элементов:

$$q = \{q_1, q_2, \dots, q_k\},$$

где  $k = \lceil \log_2(r+1) \rceil$  при  $S = \{Q_0, Q_1, \dots, Q_r\}$ , а функции переходов и выходов  $A$  и  $B_i$  реализуются с помощью комбинационных схем.

2. Управляющий автомат с *хранимой в памяти логикой* (с «запоминаемой или программируемой логикой»). Каждой выполняемой в операционном устройстве операции ставится в соответствие совокупность хранимых в памяти слов — микрокоманд, содержащих каждая информацию о микрооперациях, подлежащих выполнению в течение одного машинного такта, и указание (в общем случае зависящее от значений входных сигналов), какое должно быть выбрано из памяти следующее

слово (следующая микрокоманда). Таким образом, в этом случае функции переходов и выходов  $A$  и  $B_i$  управляющего автомата реализуются хранимой в памяти совокупностью микрокоманд.

Последовательность микрокоманд, выполняющих одну машинную команду или отдельную процедуру, образует микропрограмму. Обычно микропрограммы хранятся в специальной памяти микропрограмм (управляющей памяти).

В управляющих автоматах с хранимой в памяти программой микропрограммы используются в явной форме, они программируются в кодах микрокоманд и в таком виде заносятся в память. Поэтому такой метод управления цифровым устройством называется *микропрограммированием*, а использующие этот метод управляющие блоки — *микропрограммными управляющими устройствами*.

## 8.2. Принцип действия управляющего автомата с хранимой в памяти логикой. Микропрограммное управление

Одним из достоинств микропрограммных устройств управления является их наглядность, облегчающая изучение процесса функционирования ЭВМ и их эксплуатацию. Поэтому рассмотрение методов построения устройств управления начнем с микропрограммных устройств, не учитывая того обстоятельства, что они стали применяться позже управляющих устройств с жесткой (схемной) логикой. В настоящее время микропрограммное управление является наиболее распространенным методом построения управления, по крайней мере, в процессорах машин малой и средней производительности, а также в других устройствах (каналах, устройствах управления периферийными устройствами и др.). Микропрограммное управление применяется в некоторых типах микропроцессоров.

Хранимая в памяти микропрограмма должна содержать информацию о функциях переходов и выходов управляющего микропрограммного автомата.

Интерпретируя рассматриваемый ниже управляющий автомат (УА) в терминах конечных автоматов, обнаружим, что он функционирует подобно автомату Мили с задержанными на такт выходными сигналами. Аргументами функций переходов и выходов автомата являются входные переменные  $Z(t)$  и  $U(t)$  и переменные  $q(t)$ , задающие своими значениями состояние автомата  $Q(t)$ . Набор значений аргументов удобно отождествить с адресом микрокоманды. Этот адрес заносится в регистр адреса микрокоманды во время такта. В управляющей памяти

(УП) по данному адресу хранится код, задающий набор значений выходных сигналов управляющего автомата  $V(t+1)$  (операционная часть микрокоманды) — и набор значений переменных  $q(t+1)$ , представляющий состояние  $Q(t+1)$ . Значения  $V(t+1)$  и  $q(t+1)$  определяются функциями переходов и выходов.

Структурная схема простейшего варианта управляющего автомата с хранимой в памяти программой приведена на рис. 8.1. Автомат работает следующим образом. Серия синхросигналов  $CC$  определяет такты работы автомата, при этом значение  $CC=1$  выделяет такт, а значение  $CC=0$  — паузу между тактами. Напомним, что значения входных и выходных сигналов и состояние автомата должны быть неизменными во время такта и могут меняться только в паузах.

Состояние автомата  $Q(t)$  представляется набором значений переменных  $q(t)$ .

Пусть в такте  $t$  в  $P_2AMK$  занесены  $U(t)$ ,  $Z(t)$  и  $q(t)$ , а в  $P_2MK$  находится  $V(t)$ . Тогда в паузе перед тактом  $(t+1)$  при  $CC=0$  на  $P_2AMK$  эти значения сохраняются и из УП можно выбрать  $V(t+1)$ , которые у рассматриваемого автомата, как и  $q(t+1)$ , зависят от  $q(t)$ ,  $Z(t)$  и  $U(t)$ . Эти значения при  $CC=0$  заносятся в  $P_2MK$  (одновременно происходит изменение значений входных сигналов). После возникновения  $CC=1$ , задающего такт  $(t+1)$ , в  $P_2MK$  хранятся сформированные ранее  $V(t+1)$  и  $q(t+1)$ , при этом сигналы  $V(t+1)$  используются для инициирования микроопераций, а  $q(t+1)$  переносится в  $P_2AMK$ , после чего цикл работы автомата повторяется.

Воздействие управляющих сигналов  $V(t)$  на операционный блок синхронизируется сигналом  $CC=1$ , обеспечивающим восприятие  $V(t)$  строго в такте  $t$  из  $P_2MK$ , находящегося в режиме хранения.

Управляющая память может быть двух типов: постоянная и с произвольным обращением, т. е. допускающая как считывание, так и запись. В последнем случае загрузка УП производится пользователем со специального внешнего ЗУ по шине загрузки управляющей памяти  $ШЗгУП$  при каждом включении машины в работу.

Анализируя рассматриваемую структурную схему, можно заметить, что число слов, хранимых в УП, очень велико из-за большой разрядности  $P_2AMK$ , обусловленной значительным числом используемых оповещательных сигналов  $U(t)$ . Сократить емкость УП можно, если учесть, что для каждого  $q(t)$  существенными являются значения не всех, а лишь некоторых переменных из  $Z(t)$  и  $U(t)$ , задающих различные переходы из состояния  $Q(t)$  в состояние  $Q(t+1)$ .

Число таких переходов в микропрограммных автоматах обычно невелико, и поэтому для каждого  $Q(t)$  можно выделить группу адресов УП, хранящих коды только различных состояний перехода  $Q(t+1)$ . Емкость УП при этом во много раз сократится, так как в УП не будут храниться многократно повторяющиеся коды, описывающие одинаковые переходы автомата.

Адрес микрокоманды при таком подходе формируется специальной комбинационной схемой формирования адреса микрокоманд  $СхФАМк$  по значениям  $q(t)$ ,  $U(t)$  и  $Z(t)$ . Схема  $СхФАМк$  подключается ко входам  $РгАМк$ , как показано штриховыми линиями на рис. 8.1.

Адрес очередной микрокоманды можно назначить без учета значений  $Z(t)$  и  $U(t)$ , если эта микрокоманда задает функцию перехода автомата в состоянии, имеющем единственный переход, не зависящий от значения входных сигналов. В этом случае адрес очередной микрокоманды можно указать значением отдельной группы разрядов исполняемой микрокоманды. Если очередная микрокоманда должна задавать функцию перехода автомата в состоянии, имеющем различные переходы, зависящие от значений входных сигналов, то ее адрес должен зависеть от входных сигналов.

Для получения простой схемы  $СхФАМк$  обычно используется следующий способ формирования очередного адреса. В микрокоманде выделяется помимо операционной части адресная. Адресная часть содержит несколько полей (групп разрядов): поле типа формирования адреса (ТФА) и поля формирования отдельных групп разрядов очередного адреса (ПФА). При определенных значениях поля ТФА очередной адрес формируется только из значений ПФА (в простейшем случае в  $РгАМк$  переносятся значения ПФА, заданные в микрокоманде).

Если адрес очередной микрокоманды должен формироваться с учетом значений входных сигналов, то в поле ТФА заносится специальный код, настраивающий  $СхФАМк$  на особую обработку ПФА, при этом содержимое некоторых ПФА по-прежнему

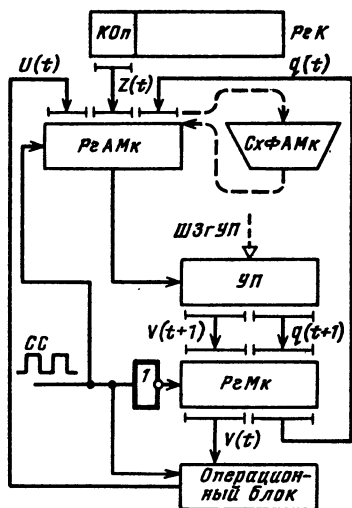


Рис. 8.1. Структурная схема простейшего варианта управляющего автомата с хранимой в памяти логикой

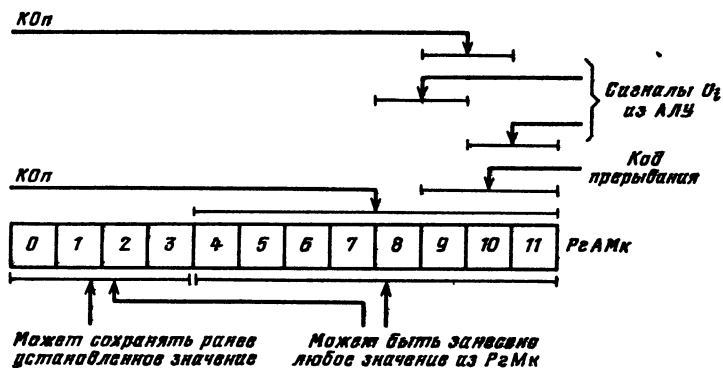


Рис. 8.2. Пример структуры PzAMk

переносится в *PzAMk*, а коды из других ПФА обеспечивают занесение в *PzAMk* значений, указываемых этими ПФА входных переменных. Таким образом, очередной адрес оказывается зависящим не только от ранее исполнявшейся микрокоманды (состояния автомата), но и от значений входных сигналов, влияющих на переходы автомата в новое состояние.

На рис. 8.2 показан пример формирования адреса микрокоманды в *PzAMk*.

Регистр *PzAMk* состоит из 12 разрядов. В разряды 0—3 и 4—11 *PzAMk* можно записывать содержимое определенных полей *PzMk*, что обеспечивает возможность задания любого адреса УП без учета значений входных сигналов. При выполнении микрокоманд, формирующих очередной адрес с учетом значений входных сигналов, в *PzAMk* можно заносить в разряды 4—11 код операции, в разряды 9—10 отдельные разряды *Коп*, в разряды 9—11 код прерывания, в разряды 8, 9 и 10, 11 значения различных оповещающих сигналов. Пользуясь этим при составлении микропрограмм, можно организовывать последовательность выполнения микрокоманд, зависящую от значений входных сигналов управляющего автомата. Для этого необходимо, кодируя каждую микрокоманду, задавать способ формирования адреса следующей микрокоманды.

Рассмотренный способ формирования адреса следующей микрокоманды носит название *принудительного формирования адреса*.

Формирование адреса следующей микрокоманды иногда осуществляют *способом естественной адресации* с помощью счетчика. Безусловный переход от микрокоманды с адресом *i* осущес-

ствляется к микрокоманде с адресом  $i+1$ , что позволяет иметь в микрокоманде только операционную часть. Однако при этом в микропрограмму вводят помимо операционных адресные микрокоманды, различаемые по специальному признаку. В адресных микрокомандах отсутствует операционная часть и все разряды используются в качестве полей ТФА и ПФА, таких же, как и в адресной части рассматривавшихся выше микрокоманд. Такой способ формирования адреса микрокоманды вызывает усложнение схем дешифрования микрокоманд и увеличение длины микропрограмм, но сокращает длину микрокоманд.

Управляющие автоматы с хранимой в памяти логикой различаются по способу формирования управляющих функциональных сигналов. Возможно использование горизонтального, вертикального и смешанного микропрограммирования.

*Горизонтальное микропрограммирование.* Каждому разряду операционной части микрокоманды ставится в соответствие определенный управляющий функциональный сигнал, т. е. определенная микрооперация. Если в разряде стоит 1, то соответствующая микрооперация выполняется независимо от значения других разрядов. При таком способе операционная часть микрокоманды содержит  $m$  разрядов, где  $m$  — общее число микроопераций. Достоинствами горизонтального микропрограммирования являются возможность одновременного выполнения в одном такте любого набора из  $m$  микроопераций и простота формирования функциональных сигналов, так как последние могут возбуждаться непосредственно от сигналов из регистра микрокоманды.

Однако оно имеет и существенный недостаток, заключающийся в том, что требуется большая длина микрокоманды, поскольку число функциональных сигналов в современном процессоре может достигать нескольких сотен. Поэтому, хотя и известны случаи практического применения горизонтального микропрограммирования, главным образом в малых машинах, где число управляющих функциональных сигналов сравнительно невелико (около 150), большее распространение получили другие методы.

При *вертикальном микропрограммировании* микрооперация определяется не состоянием одного из разрядов микрокоманды, а двоичным кодом, содержащимся в операционной части микрокоманды, при этом отдельный код задает отсутствие микрооперации.

Число разрядов операционной части микрокоманды

$$n_{o.ч} = \lceil \log_2 (m + 1) \rceil.$$



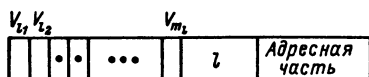


Рис. 8.3. Структура микрокоманды при вертикально-горизонтальном микропрограммировании

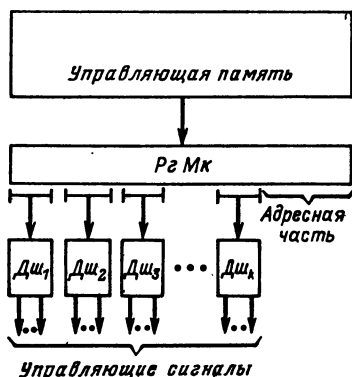


Рис. 8.4. Структура управляющего автомата при горизонтально-вертикальном микропрограммировании

Достоинством вертикального микропрограммирования является небольшая длина микрокоманды. Однако в этом случае требуются сложные дешифраторы на большое число микроопераций, а главное — в каждой микрокоманде указывается лишь одна микрооперация, что приводит к увеличению длины микропрограмм по сравнению с их длиной при горизонтальном микропрограммировании. Вертикальное микропрограммирование часто используется в микропрограммируемых микропроцессорах, как, например, для управления центральными процессорными элементами набора K1802.

В настоящее время наибольшее распространение имеет *смешанное микропрограммирование*, в котором сочетаются горизонтальное и вертикальное микропрограммирование.

При смешанном микропрограммировании множество микроопераций  $V$  разбивается на  $k$  подмножеств (или полей):

$$V = \bigcup_{l=1}^k V_l.$$

Микрооперации внутри каждого из подмножеств кодируются либо горизонтальным, либо вертикальным способом. Можно выделить два способа.

*Вертикально-горизонтальное микропрограммирование.* Все множество микроопераций  $V$  расчлняют на  $k$  подмножеств  $V_l$ , в каждом из которых объединяют микрооперации, наиболее часто встречающиеся вместе в одном такте. Подмножества стараются по возможности сделать равномошными. Операционная часть микрокоманды (рис. 8.3) состоит из двух полей. В первом поле, длина которого равна  $\max |V_l|$ , применен горизонтальный способ кодирования микроопераций, т. е. каждый разряд соответствует определенной микрооперации из подмножества  $V_l$ ,

а другое поле, имеющее длину  $\lceil \log_2 k \rceil$ , указывает, к какому из  $k$  подмножеств принадлежат микрооперации в первом поле микрокоманды.

Более гибким и часто используемым является горизонтально-вертикальный способ кодирования.

*Горизонтально-вертикальное микропрограммирование* (рис. 8.4). Подмножества  $V_i$  кодируются горизонтальным, а микрооперации внутри каждого из подмножеств — вертикальными способами. В этом случае каждому подмножеству  $V_i$  выделяется отдельное поле в операционной части микрокоманды. Длина операционной части микрокоманды

$$n_{o.v} = \sum_{i=1}^k \lceil \log_2 (m_i + 1) \rceil,$$

где  $m_i$  — число микроопераций, представляемых в поле  $i$ .

Микрокоманды такого типа называют микрокомандами с полевой структурой. При таком способе кодирования желательно, чтобы

$$V_i \cap V_j = \emptyset \text{ при } i \neq j,$$

т. е. чтобы каждая микрооперация встречалась только в одном поле.

При *прямом кодировании микрокоманд* каждое поле микрокоманды несет фиксированные функции. *Косвенное кодирование* характеризуется наличием дополнительных полей, содержимое которых меняет смысл основных полей микрокоманды. Таким образом, интерпретация полей, формирующих управляющие сигналы, зависит от бит дополнительных полей. (Подобный подход используется в вертикально-горизонтальном микропрограммировании.)

Косвенное кодирование широко используется, так как позволяет уменьшить длину микрокоманды. Однако оно в некоторой степени нарушает стройность микропрограммного управления, вызывает усложнение дешифраторов и приводит к снижению скорости работы из-за потерь времени на дешифрирование дополнительных полей микрокоманды.

Различают *одно- и многофазные микрокоманды*. В первом случае все микрооперации, указанные в микрокоманде, выполняются одновременно в течение одного такта. Во втором такт разбивается на части, называемые фазами или микротактами, и указанные в микрокоманде микрооперации выполняются в различные микротакты (фазы такта). В этом случае приходится учитывать временные зависимости между отдельными микрооперациями. Однако становится возможным включать в микрокоманду взаимно исключающие микрооперации, разводя их по

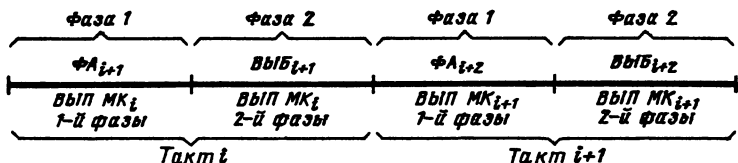


Рис. 8.5. Временная диаграмма работы управляющего автомата с хранимой в памяти логикой

разным тактам. Многофазные микрокоманды часто применяются в ЭВМ с горизонтальным или горизонтально-вертикальным микропрограммированием (например, в ЕС ЭВМ).

Для повышения скорости работы УА с хранимой в памяти логикой используются различные методы. Чаще всего осуществляется совмещение отдельных действий, задаваемых микрокомандой, что особенно существенно при использовании многофазных микрокоманд. Например, двухфазные микрокоманды могут выполняться с совмещениями вида, указанного на рис. 8.5. В такте  $i$  работы осуществляется выполнение микроопераций  $i$ -й микрокоманды  $ВЫП МК_i$ , а одновременно с этим происходит формирование адреса очередной микрокоманды  $ФА_{i+1}$  и ее выборка  $ВЫБ_{i+1}$ . При таком совмещении адрес  $(i+1)$ -й микрокоманды формируется  $i$ -й микрокомандой с учетом значений оповещающих сигналов, представляющих собой результаты  $(i-1)$ -й микрокоманды. Учет результатов  $i$ -й микрокоманды может осуществляться только  $(i+1)$ -й микрокомандой при формировании адреса  $(i+2)$ -й микрокоманды.

Хотя идея микропрограммирования известна с 1951 г. [125], однако довольно долго этот принцип управления не находил широкого применения в ЭВМ из-за отсутствия достаточно надежных и дешевых быстродействующих УП для хранения микропрограмм и сложности разработки текстов микропрограмм. Однако в последние годы вырос интерес к микропрограммному принципу управления (к УА с хранимой в памяти логикой). Созданы постоянные запоминающие устройства для УП с циклом обращения 0,25—0,5 мкс. Появились интегральные ПЗУ и ЗУ с произвольным обращением, обладающие еще большим быстродействием. Выяснилось, что эффективность методов построения УА зависит от числа и сложности команд ЭВМ и при увеличении и усложнении команд и функций процессора эффективность микропрограммного управления растет.

Использование принципа микропрограммного управления позволяет строить более регулярные схемы УА, создавать эффективные системы диагностики для автоматического поиска не-

исправностей, упрощает документирование алгоритмов работы УА. В последнее время микропрограммирование используется как средство для аппаратурной реализации фрагментов операционных систем, трансляторов и т. п.

Микропрограммирование широко используется для проблемной ориентации микропроцессорных устройств и систем при помощи специализированного набора команд, обеспечивающего наиболее эффективное решение определенных задач пользователя.

Все чаще в ЭВМ применяется загружаемая управляющая память, при этом в качестве первичных носителей микропрограмм используются гибкие диски или кассеты портативных магнитофонов, с которых микропрограммы загружаются в УП.

Микропрограммное управление с использованием загружаемой УП позволяет расширять или даже менять состав команд ЭВМ.

Наиболее выгодно использование УА с хранимой в памяти логикой для операционных блоков процессоров, в которых реализуются алгоритмы с относительно небольшим числом условий ветвления. Реализация в этих автоматах алгоритмов с большим числом условий ветвления ведет к значительному усложнению *СхФАМк* и, следовательно, к увеличению времени формирования адреса микрокоманды, а в конечном счете к уменьшению быстродействия УА с хранимой в памяти логикой. В этом случае используются УА с «жесткой» логикой, обладающие большим быстродействием. В некоторых случаях в ЭВМ используются одновременно УА с хранимой в памяти логикой и УА с «жесткой» логикой.

### **8.3. Управляющие автоматы с «жесткой» логикой**

Управляющие автоматы с «жесткой» логикой представляют собой логические схемы, вырабатывающие распределенные во времени управляющие функциональные сигналы. В отличие от управляющих устройств с хранимой в памяти логикой у этих автоматов можно изменить логику работы только путем переделок схем автомата.

Типичная структурная схема управляющего автомата с «жесткой» логикой показана на рис. 8.6. В состав схемы входят регистр кода операции, являющийся частью регистра команд, счетчик тактов, дешифратор тактов и дешифратор кода операции, а также логические схемы образования управляющих функциональных сигналов.

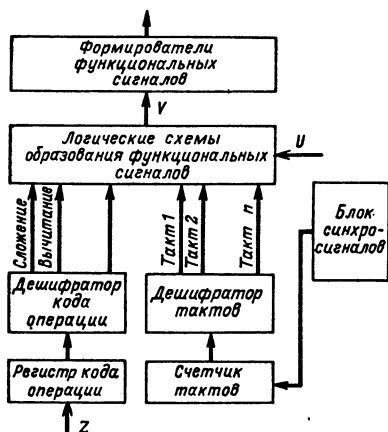


Рис. 8.6. Структура управляющего автомата с «жесткой» логикой

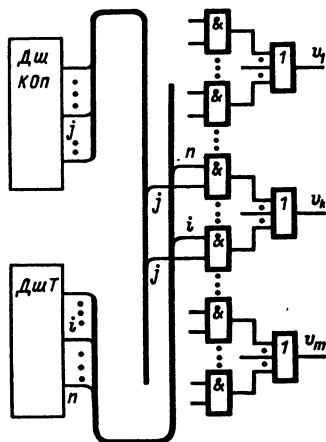


Рис. 8.7. Фрагмент схемы образования управляющих функциональных сигналов:

Дш КОП — дешифратор кода операции; Дш Т — дешифратор тактов

На счетчик тактов поступают сигналы от блока синхросигналов, и счетчик с каждым сигналом меняет свое состояние. Состояния счетчика представляют собой номера тактов, изменяющиеся от 1 до  $n$ . Дешифратор тактов формирует на  $i$ -м выходе единичный сигнал при  $i$ -м состоянии счетчика тактов, т. е. во время  $i$ -го такта.

Дешифратор кода операции вырабатывает единичный сигнал на  $j$ -м выходе, если исполняется  $j$ -я команда.

Логические схемы образования управляющих функциональных сигналов для каждой команды возбуждают формирователи функциональных сигналов для выполнения требуемых в данном такте микроопераций.

Принцип построения логических схем образования управляющих сигналов поясняется на рис. 8.7. Здесь показан фрагмент схемы, обеспечивающей выработку управляющего сигнала  $u_k$  в  $i$ -м и  $n$ -м тактах выполнения  $j$ -й команды.

В общем случае значения управляющих сигналов зависят еще и от оповещающих сигналов, отражающих ход вычислительного процесса. Для реализации этих зависимостей элементы, представленные на рис. 8.7, берутся многовыходовыми и на них заводятся требуемые сигналы логических условий.

Если, например, необходимо, чтобы при исполнении  $j$ -й команды управляющий сигнал  $u_k$  появлялся в  $i$ -м такте только при

значениях оповещающих сигналов  $u_1=0$  и  $u_3=1$ , а в  $n$ -м такте всегда, то схема, приведенная на рис. 8.7, изменится и примет вид, представленный на рис. 8.8.

Серьезным недостатком рассмотренных схем является одинаковое число тактов для всех команд. Это требует выравнивания числа тактов исполнения команд по наиболее «длинной» команде, что ведет к непроизводительным затратам времени. Чтобы устранить этот недостаток, схемы строят с использованием нескольких счетчиков тактов.

Схема формирования тактовых сигналов (датчик тактовых сигналов) может строиться на основе использования регистра сдвига, по которому двигается одна 1 (регистр с «бегущей единицей»), что не требует использования дешифратора.

Построение управляющих автоматов с «жесткой» логикой формализуется на основе интерпретации микропрограмм автоматами [7].

Работу операционного блока можно описать микропрограммой, например, на языке микроопераций или в виде графа. По микропрограмме строится соответствующий управляющий автомат типа Мура или Мили.

Рассмотрим способ построения автомата Мура. Пусть микропрограмма работы некоторого операционного блока имеет вид, представленный на рис. 8.9, а. Блок ожидает возникновения  $u_1=1$  и затем производит выработку управляющих функциональных сигналов  $v_1—v_6$  в определенной последовательности, зависящей от значений сигналов  $u_2$  и  $u_3$ .

Каждой микрокоманде, отдельно представленной на графе, ставится в соответствие отдельное состояние автомата. Состояния автомата отмечаются управляющими функциональными сигналами соответствующих микрокоманд.

Условия перехода от микрокоманды к микрокоманде представляются в виде конъюнкции входных сигналов, влияющих на переход. Каждая конъюнкция выписывается так, чтобы набор

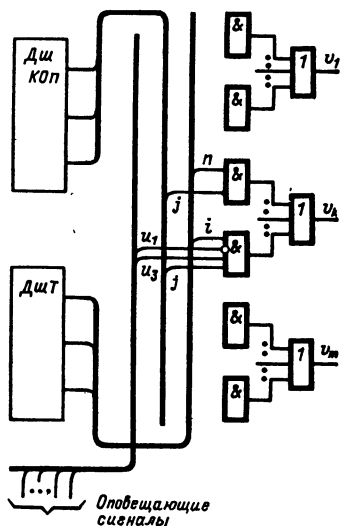


Рис. 8.8. Фрагмент 'схемы образования управляющих функциональных сигналов с реализацией зависимости от оповещающих сигналов

значений входных переменных, обращающих конъюнкцию в 1, соответствовал условию перехода. При безусловном переходе конъюнкция заменяется на константу 1.

Начало и конец микропрограммы отображаются начальным состоянием  $Q_0$  автомата Мура, при этом предполагается, что автомат является циклическим, т. е. допускающим многократную повторяющуюся выработку последовательностей управляющих сигналов.

Граф автомата Мура, построенного по описанным правилам для рассматриваемой микропрограммы, представлен на рис. 8.9, б.

Микропрограмму можно интерпретировать не только автоматом Мура, но и автоматом Мили, отличающимся тем, что значение его функции выхода в момент  $t$  зависит не только от состояния автомата, но и от набора значений входных сигналов.

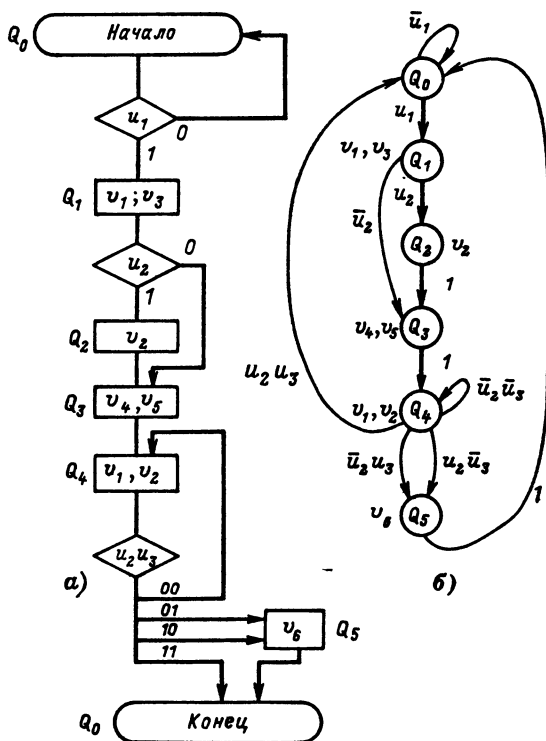


Рис. 8.9. Микропрограмма, представленная графом (а), и граф автомата Мура (б), интерпретирующей микропрограмму

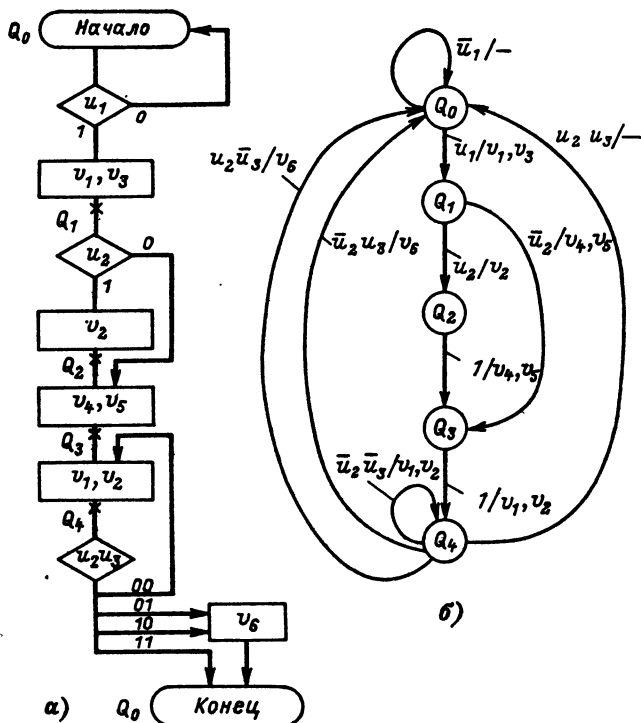


Рис. 8.10. Микропрограмма (а) и граф автомата Мили (б), интерпретирующей микропрограмму

Переход от микропрограммы к автомату Мили иллюстрируется на рис. 8.10, на котором показаны рассмотренный выше граф микропрограммы и граф автомата Мили, интерпретирующей ее. Начало и конец микропрограммы представляются начальным состоянием автомата  $Q_0$ . Каждая дуга, выходящая из прямоугольника, представляющего собой микрокоманду, отмечается меткой  $\times$  и символом состояния автомата. Исключение составляют дуги, идущие к конечной или к начальной вершине, которые не отмечаются. Если несколько дуг с меткой  $\times$  входят в один блок графа микропрограммы, то все они отмечаются одинаковым символом состояния.

Условия перехода по микропрограмме от одной метки состояния к другой, соседней, задают функцию переходов автомата. Эти условия записываются в виде конъюнкций так же, как это делалось для автоматов Мура. Для каждого перехода, кроме



того, фиксируется набор выходных переменных, принимающих при переходе единичное значение (задание функции выходов).

Автомат Мили, построенный по микропрограмме, имеет число состояний, как правило, меньшее числа состояний эквивалентного ему автомата Мура. С этой точки зрения использование автомата Мили предпочтительно. Однако применение автомата Мили в качестве управляющего автомата не всегда возможно. Объясняется это тем, что УА работает в контуре с операционным блоком ОБ (рис. 8.11). У автомата Мили переход в новое состояние осуществляется одновременно с формированием выходного сигнала. Поэтому, если ОБ вырабатывает оповещающие сигналы  $u_1, \dots, u_n$  сразу при возникновении управляющих сигналов, а управляющий автомат является автоматом Мили, возможна следующая недопустимая ситуация: автомат Мили еще не сменил состояние, а на его входы пришли новые значения оповещающих сигналов, требующие выполнения иного перехода.

Для исключения возможных сбоев в работе управляющих автоматов ставятся специальные схемы задержки, или, что то же самое, один из двух автоматов (управляющий либо операционный) выполняют в виде автомата Мура либо автомата Мили с задержкой, который меняет выходной сигнал после смены состояния (перехода).

Рассмотрим, каким образом, имея описание управляющего автомата, можно построить схему автомата.

Зная число состояний автомата  $r$ , определяем число элементарных автоматов (двухступенчатых триггеров), требуемых для представления состояний автомата. Это число равно  $k = \lfloor \log_2 r \rfloor$ . Затем поставим в соответствие наборы значений состояний триггеров состояниям автомата. Пусть, например, нуле-

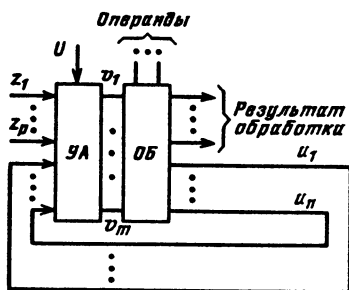


Рис. 8.11. Взаимодействие управляющего автомата и операционного блока

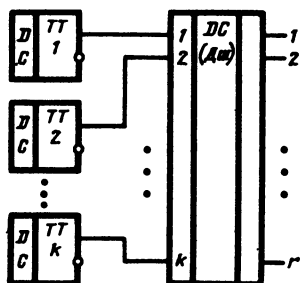


Рис. 8.12. Схема выделения состояний управляющего автомата

вое состояние всех триггеров представляет собой  $Q_0$ , набор состояний триггеров  $00\dots001$  соответствует  $Q_1$  и т. д. Далее можно построить схему из  $k$  триггеров и дешифратора с  $r$  выходами. Единичный сигнал на одном из выходов дешифратора указывает соответствующее состояние автомата. Выбрав для определенности  $D$ -триггеры, получим схему выделения состояний управляющего автомата, показанную на рис. 8.12.

Для реализации схемы формирования управляющих сигналов следует выписать для каждого управляющего сигнала его булеву функцию. Для автомата Мура эти функции (функции выходов) имеют вид  $v_i = \bigvee_j Q_{ij} = Q_{i1} \vee Q_{i2} \vee \dots \vee Q_{i_w}$ , где  $Q_{ij}$  — состояние автомата, отмеченное  $v_i$ .

Так, для автомата, представленного на рис. 8.9, имеем следующие функции выходов:

$$v_1 = Q_1 \vee Q_4; \quad v_2 = Q_2 \vee Q_4;$$

$$v_3 = Q_1; \quad v_4 = Q_3;$$

$$v_5 = Q_3; \quad v_6 = Q_5.$$

По этим функциям легко строится схема формирования управляющих сигналов (рис. 8.13, а). Выдача выходных сигналов синхронизируется так, чтобы она происходила в то время, когда сигналы не меняются.

Для автомата Мили функции выхода имеют вид  $v_i = \bigvee_j (Q_{ij} (\bigvee K_j))$ , где  $Q_{ij}$  — состояние, из которого выходит дуга, отмеченная выходным сигналом  $v_i$ , а  $\bigvee K_j$  — дизъюнкция всех конъюнкций, записанных на дугах, выходящих из  $Q_{ij}$  и отмеченных выходным сигналом.

Для автомата, представленного на рис. 8.10, функции выходов имеют вид

$$v_1 = Q_0 (u_1) \vee Q_3 \vee Q_4 (\bar{u}_2 \bar{u}_3);$$

$$v_2 = Q_1 (u_2) \vee Q_3 \vee Q_4 (\bar{u}_2 \bar{u}_3); \quad v_3 = Q_0 (u_1);$$

$$v_4 = Q_1 (\bar{u}_2) \vee Q_2; \quad v_5 = Q_1 (\bar{u}_2) \vee Q_2;$$

$$v_6 = Q_4 (\bar{u}_2 u_3 \vee u_2 \bar{u}_3).$$

Преобразовав выражения с учетом общих частей формул, получим более простые зависимости:

$$v_1 = v_3 \vee \varphi; \quad v_2 = Q_1 (u_2) \vee \varphi; \quad v_3 = Q_0 (u_1);$$

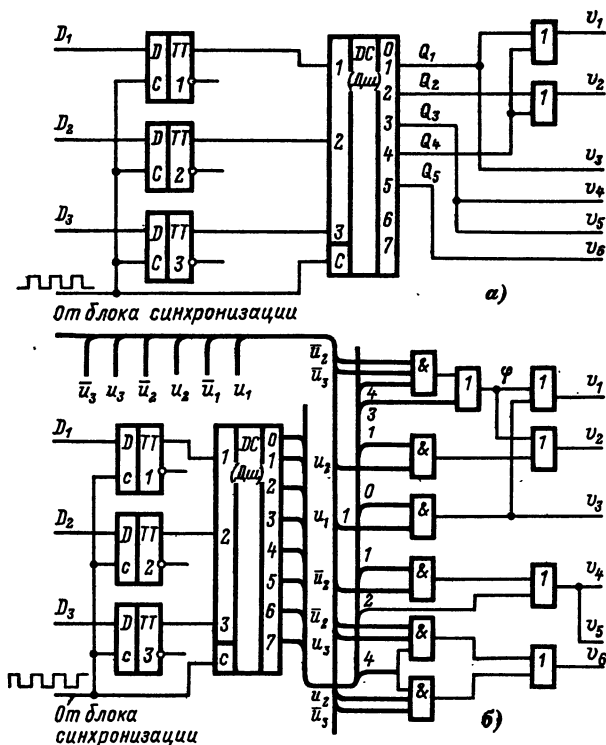


Рис. 8.13. Схема формирования управляющих сигналов для автоматов Мура (а) и Мили (б)

$$v_4 = v_5; \quad v_5 = Q_1 (\bar{u}_2) \vee Q_2;$$

$$v_6 = Q_4 (\bar{u}_2 u_3) \vee Q_4 (u_2 \bar{u}_3); \quad \varphi = Q_3 \vee Q_4 (\bar{u}_2 \bar{u}_3).$$

По этим выражениям строим схему, представленную на рис. 8.13, б. Выходные сигналы здесь синхронизированы, как и в схеме на рис. 8.13, а.

Автомат содержит схему, реализующую смену состояний триггеров в соответствии с функцией переходов.

При смене состояний автомата  $Q_i \rightarrow Q_j$  происходит смена состояний отдельных триггеров, так как они представляют собой состояния автомата. Следовательно, для каждого триггера известно, какими должны быть изменения его состояний на всех переходах. Зная тип триггера, можно определить, какие сигналы должны быть поданы на его входы для получения требуемой смены состояний.

Для каждого перехода (указаны на графе автомата дугами) выписываются исходное состояние перехода, набор исходных состояний триггеров, набор состояний триггеров после перехода, а также конъюнкции входных сигналов, представляющие собой условие перехода.

По наборам исходных состояний и состояний после перехода выделяются отдельно для каждого триггера ситуации, при которых значения его входов должны быть единичными. Описав все случаи, требующие выработки единичного сигнала, булевыми функциями состояний автомата и входных сигналов, получаем так называемые *функции возбуждения*, по которым строятся схемы формирования сигналов на входах триггеров.

Поясним сказанное примером. Пусть состояния автомата, заданного графом на рис. 8.9, представлены следующими наборами состояний триггеров (табл. 8.1).

Тогда функции переходов можно представить табл. 8.2, по которой строим булевы функции возбуждения  $D_1$ ,  $D_2$  и  $D_3$ :

$$D_1 = Q_3 \vee Q_4 \bar{u}_2 \bar{u}_3 \vee Q_4 \bar{u}_2 u_3 \vee Q_4 u_2 \bar{u}_3;$$

$$D_2 = Q_1 u_2 \vee Q_1 \bar{u}_2 \vee Q_2;$$

$$D_3 = Q_0 u_1 \vee Q_1 \bar{u}_2 \vee Q_2 \vee Q_4 \bar{u}_2 u_3 \vee Q_4 u_2 \bar{u}_3.$$

Упростив выражения, получим

$$D_1 = Q_3 \vee Q_4 \bar{u}_2 \vee Q_4 \bar{u}_3; \quad D_2 = Q_1 \vee Q_2;$$

$$D_3 = Q_0 u_1 \vee Q_1 \bar{u}_2 \vee Q_2 \vee Q_4 \bar{u}_2 u_3 \vee Q_4 u_2 \bar{u}_3.$$

По упрощенным выражениям строим схему (рис. 8.14).

Объединив схему формирования выходных сигналов (см. рис. 8.13, а) и схему формирования функций возбуждения (рис. 8.14), получим схему, реализующую автомат, представленный на рис. 8.9.

Для получения более экономичных схем выделение состояний автомата часто делается неполным, при этом триггеры

Т а б л и ц а 8.1

Состояние автомата	Состояние триггера			Состояние автомата	Состояние триггера		
	$T_{z_1}$	$T_{z_2}$	$T_{z_3}$		$T_{z_1}$	$T_{z_2}$	$T_{z_3}$
$Q_0$	0	0	0	$Q_3$	0	1	1
$Q_1$	0	0	1	$Q_4$	1	0	0
$Q_2$	0	1	0	$Q_5$	1	0	1

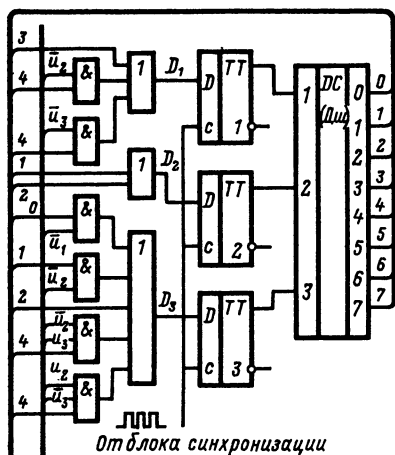


Рис. 8.14. Схема формирования функций возбуждения

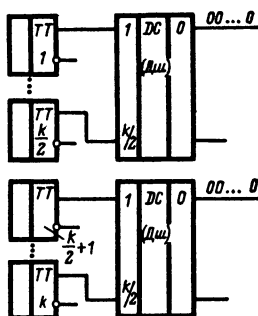


Рис. 8.15. Схема неполного выделения состояний автомата

разделяются на две или более группы, каждая из которых связывается с отдельным дешифратором.

Число выходов схемы неполного выделения состояний резко сокращается и составляет  $s = 2^{(k/l)+1}$ , где  $l$  — число групп по  $k/l$  триггеров. Пример такой схемы с  $l=2$  представлен на рис. 8.15.

Для полного выделения состояния в этом случае необходимо сигналы с выходов отдельных дешифраторов подать на элемент И. Это можно делать непосредственно в схемах формиро-

Таблица 8.2

Исходное состояние	Исходное состояние триггера			Состояние триггера после перехода			Условие перехода	Входной сигнал, вызывающий нужный переход триггера		
	$T_{z1}$	$T_{z2}$	$T_{z3}$	$T_{z1}$	$T_{z2}$	$T_{z3}$		$D_1$	$D_2$	$D_3$
$Q_0$	0	0	0	0	0	0	$\bar{u}_1$	0	0	0
$Q_0$	0	0	0	0	0	1	$u_1$	0	0	1
$Q_1$	0	0	1	0	1	0	$u_2$	0	1	0
$Q_1$	0	0	1	0	1	1	$\bar{u}_2$	0	1	1
$Q_2$	0	1	0	0	1	1	1	0	1	1
$Q_3$	0	1	1	1	0	0	1	1	0	0
$Q_4$	1	0	0	1	0	0	$\bar{u}_2 \bar{u}_3$	1	0	0
$Q_4$	1	0	0	1	0	1	$u_2 u_3$	1	0	1
$Q_4$	1	0	0	1	0	1	$u_2 \bar{u}_3$	1	0	1
$Q_4$	1	0	0	0	0	0	$u_2 u_3$	0	0	0
$Q_5$	1	0	1	0	0	0	1	0	0	0

вания выходных сигналов и функций возбуждения, что дает в итоге более экономную общую схему.

При синтезе схем автоматов формальным методом приходится решать вопросы оптимального кодирования состояний, минимизации числа состояний автоматов, факторизации выражений, описывающих булевы функции возбуждения и выхода, и ряд других. Подробно методика формального синтеза схем управляющих автоматов описана в [7].

#### 8.4. Программируемые логические матрицы в управляющих автоматах

Логическим элементом, весьма удобным для использования в схемах управляющих автоматов, является интегральная микросхема, называемая *программируемой логической матрицей* (ПЛМ). Схема ПЛМ имеет вид, представленный на рис. 8.16.

При изготовлении ПЛМ образуется схема, допускающая множество вариантов обработки входных сигналов. Входные элементы позволяют иметь все входные переменные как в прямой, так и в инверсной форме. На входы любого элемента И поданы все входные переменные и их инверсии. Ко входам каждого элемента ИЛИ подключены выходы всех элементов И. Наконец, выходные элементы позволяют получить любую из выходных функций в прямом или инверсном виде.

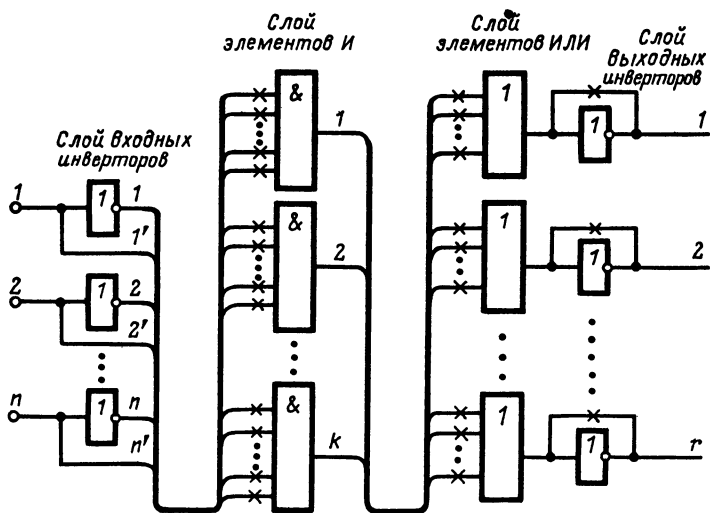


Рис. 8.16. Схема ПЛМ

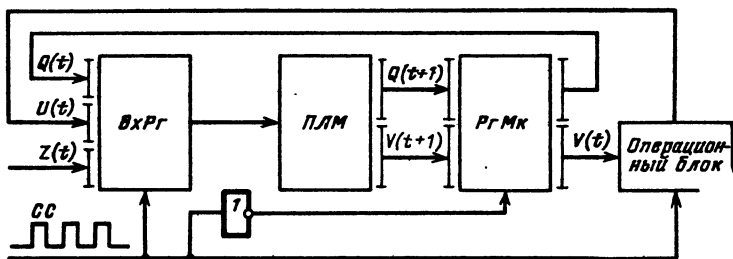


Рис. 8.17. Использование ПЛМ в управляющем автомате

Программирование матрицы состоит в устранении ненужных связей (устраиваемые связи обозначены крестиками на рис. 8.16): Устранение лишних связей производится с помощью фототаблонов или выжиганием.

Программируя ПЛМ, можно реализовать нужные системы булевых функций. Это позволяет строить управляющие автоматы со структурой, представленной на рис. 8.17. Функционирует эта схема аналогично схеме на рис. 8.1. Применение СхФАМк здесь обычно не является необходимым из-за большого числа входов ПЛМ. Часть выходов ПЛМ используется для выработки управляющих функциональных сигналов, а часть — для выработки сигналов функций возбуждения. При этом часто оказывается необходимой совместная минимизация реализуемой ПЛМ системы булевых функций. Именно совместная минимизация дает возможность «заложить» в ПЛМ достаточно сложные функции.

Очень удобны для построения управляющих автоматов микросхемы, содержащие в одном корпусе ПЛМ с набором выходных триггеров.

### Контрольные вопросы и упражнения

1. Назовите два основных типа управляющих автоматов и поясните их различия.
2. С какой целью в УА с хранимой в памяти логикой вводится схема формирования адреса микрокоманды?
3. Сравните принудительный и естественный способы адресации микрокоманд.
4. Чем отличается вертикально-горизонтальное микропрограммирование от горизонтально-вертикального?
5. В чем отличие правил построения графа автомата Мили от правил построения графа автомата Мура?

6. Почему автомат Мили не всегда может использоваться в качестве управляющего автомата?

7. Почему при построении УА на базе ПЛМ не строят схему выделения состояний, подобную схемам на рис. 8.12 и 8.15?

## Глава 9

# ПРОЦЕССОРЫ И МИКРОПРОЦЕССОРЫ: ЭЛЕМЕНТЫ АРХИТЕКТУРЫ

## 9.1. Предварительные замечания

Выделим следующие элементы архитектуры ЭВМ:  
типы обрабатываемых в ЭВМ данных и способы их представления в машине (изложены в гл. 2);  
адресные структуры основной памяти;  
способы адресации информации;  
структуры команд и их микропрограммная интерпретация;  
регистровые структуры (программистские модели) процессоров;  
системы прерывания;  
особенности систем команд ЭВМ;  
рабочий цикл процессора;  
конвейеризация обработки команд и данных и др.

В настоящей главе с единых позиций рассматриваются элементы архитектуры ЭВМ общего назначения (ЕС ЭВМ), малых ЭВМ (СМ ЭВМ), микроЭВМ и микропроцессоров.

## 9.2. Назначение и структура процессора

*Процессором* называется устройство, непосредственно осуществляющее процесс обработки данных и программное управление этим процессом. Процессор дешифрирует и выполняет команды программы, организует обращения к оперативной памяти, в нужных случаях инициирует работу периферийных устройств, воспринимает и обрабатывает запросы, поступающие из устройств машины и из внешней среды («запросы прерывания»).

Процессор занимает центральное место в структуре ЭВМ, так как он осуществляет управление взаимодействием всех устройств, входящих в состав ЭВМ.



Выполнение команды (машинной операции) разделено на более мелкие этапы — микрооперации (микрокоманды), во время которых выполняются определенные элементарные действия. Конкретный состав микроопераций определяется системой команд и логической структурой данной ЭВМ. Последовательность микроопераций (микрокоманд), реализующих данную операцию (команду), образует микропрограмму операции.

Для определения временных соотношений между различными этапами операции используется понятие *машинного такта*. Машинный такт определяет интервал времени, в течение которого выполняется одна или одновременно несколько микроопераций. Границы тактов задаются синхросигналами, вырабатываемыми специальной схемой — генератором синхросигналов.

Таким образом, может быть установлена следующая иерархия этапов выполнения программ в процессоре: программа, команда (микропрограмма), микрооперация (микрокоманда).

Упрощенная структурная схема процессора представлена на рис. 9.1. На схеме изображены только его основные части: арифметическо-логическое устройство *АЛУ*, управляющее устройство (управляющий автомат) *УУ*, блок управляющих регистров *БУР*, блок регистровой памяти (*местная память*) и блок связи с ОП и некоторым другим, в том числе внешним по отношению к ЭВМ, оборудованием.

В состав процессора могут также входить и некоторые другие блоки, участвующие в организации вычислительного процесса (блок прерывания, блок защиты памяти, блок контроля правильности работы и диагностики процессора и др.).

Оперативная (основная) память выполняется в виде отдельного устройства, хотя в небольших ЭВМ может конструктивно объединяться с процессором и использовать частично его оборудование.

Арифметическо-логическое устройство процессора выполняет логические и арифметические операции над данными. В общем случае в АЛУ выполняются логические преобразования над логическими кодами фиксированной и переменной длины (над отдельными битами, группами бит, байтами и их последовательностями), арифметические операции над числами с фикси-

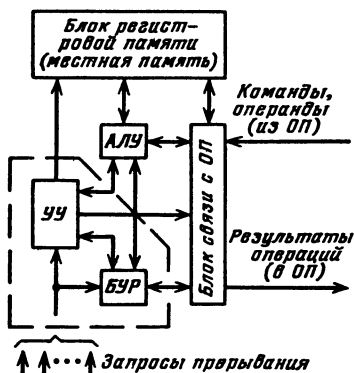


Рис. 9.1. Упрощенная структурная схема процессора

рованной и плавающей точками, над десятичными числами, обработка алфавитно-цифровых слов переменной длины и др. Характер выполняемой АЛУ операции задается командой программы.

В процессоре может быть одно универсальное АЛУ для выполнения всех основных арифметических и логических преобразований или несколько специализированных для отдельных видов операций. В последнем случае увеличивается количество оборудования процессора, но повышается его быстродействие за счет специализации и упрощения схем выполнения отдельных операций.

*Управляющее устройство* (управляющий автомат) вырабатывает последовательность управляющих сигналов, инициирующих выполнение соответствующей последовательности микроопераций, обеспечивающей реализацию текущей команды.

В процессорах ЭВМ и микропроцессорах применяют управляющие автоматы с хранимой в памяти логикой (микропрограммные управляющие устройства) и с «жесткой» логикой (см. гл. 8).

*Блок управляющих регистров* предназначен для временного хранения управляющей информации. Он содержит регистры и счетчики, участвующие в управлении вычислительным процессом: регистры, хранящие информацию о состоянии процессора, регистр-счетчик адреса команды — *счетчик команд*, счетчики тактов, регистр запросов прерывания и др.

К блоку управляющих регистров следует также отнести управляющие триггеры, фиксирующие режимы работы процессора.

Для повышения быстродействия и логических возможностей процессора и микропроцессора в их состав включают *блок регистровой памяти (местную память)* небольшой емкости, но более высокого, чем ОП, быстродействия. Регистры этого блока (или ячейки местной памяти) указываются в командах программы путем укороченной регистровой адресации и служат для хранения операндов, в качестве аккумуляторов (регистров результата операций), базовых и индексных регистров, указателя стека.

В некоторых процессорах базовые и индексные регистры входят в состав блока управляющих регистров.

Местная память выполняется главным образом в виде быстродействующих полупроводниковых интегральных ЗУ.

*Блок связи (интерфейс процессора)* организует обмен информацией процессора с оперативной памятью и защиту участков ОП от недозволённых данной программе обращений, а также связь процессора с периферийными устройствами и внешним по отношению к ЭВМ оборудованием (другими ЭВМ и т. д.).

*Блок контроля и диагностики* (на рис. 9.1 не показан) служит для обнаружения сбоев и отказов в аппаратуре процессора, восстановления работы программы после сбоев и поиска места неисправности при отказах.

Процессор снабжается *пультом* со средствами индикации и ручного управления.

В предыдущих главах были подробно изложены принципы построения и функционирования основных устройств процессора — АЛУ и управляющего устройства.

В этой главе рассматриваются главным образом вопросы, относящиеся к реализации программного управления вычислительным процессом. Ряд вопросов, относящихся к этой проблеме (защита памяти, автоматический контроль правильности работы ЭВМ и др.), рассматривается в последующих главах.

### 9.3. Адресные структуры основных памяти

Основная (оперативная) память ЭВМ обычно является адресной. Это значит, что каждой хранимой в ОП единице информации (байту, слову) ставится в соответствие специальное число — адрес, определяющий место ее хранения в памяти.

В современных ЭВМ различных типов, как правило, минимальной адресуемой в памяти единицей информации является байт, т. е. 8-разрядный код (часто с дополнительным девятым контрольным разрядом — см. гл. 12). Более крупные единицы информации — основная машинная единица информации — слово и производные — полуслово, двойное слово и т. п. — образуются из целого числа байт. Обычно слово соответствует формату данных, наиболее часто встречающихся в данной машине в качестве операнда. Часто, но необязательно, формат слова соответствует ширине выборки из основной памяти.

На рис. 9.2 представлены адресные структуры основных памяти в машинах разного типа и размещение в памяти различных единиц информации. В соответствии со сложившейся у разработчиков разных типов ЭВМ традицией в машинах общего назначения нумерация бит и байт в слове и других единицах информации производится слева направо, а в малых, микроЭВМ и микропроцессорах — справа налево. В обоих случаях адресом слова (и других единиц информации) служит адрес их байта с наименьшим номером, но в машинах общего назначения им является крайний левый (старший) байт в представлении числа (команды и т. п.), а в малых ЭВМ, микроЭВМ и микропроцессорах — крайний правый (младший) байт.

Число байт в адресуемой единице информации или задается

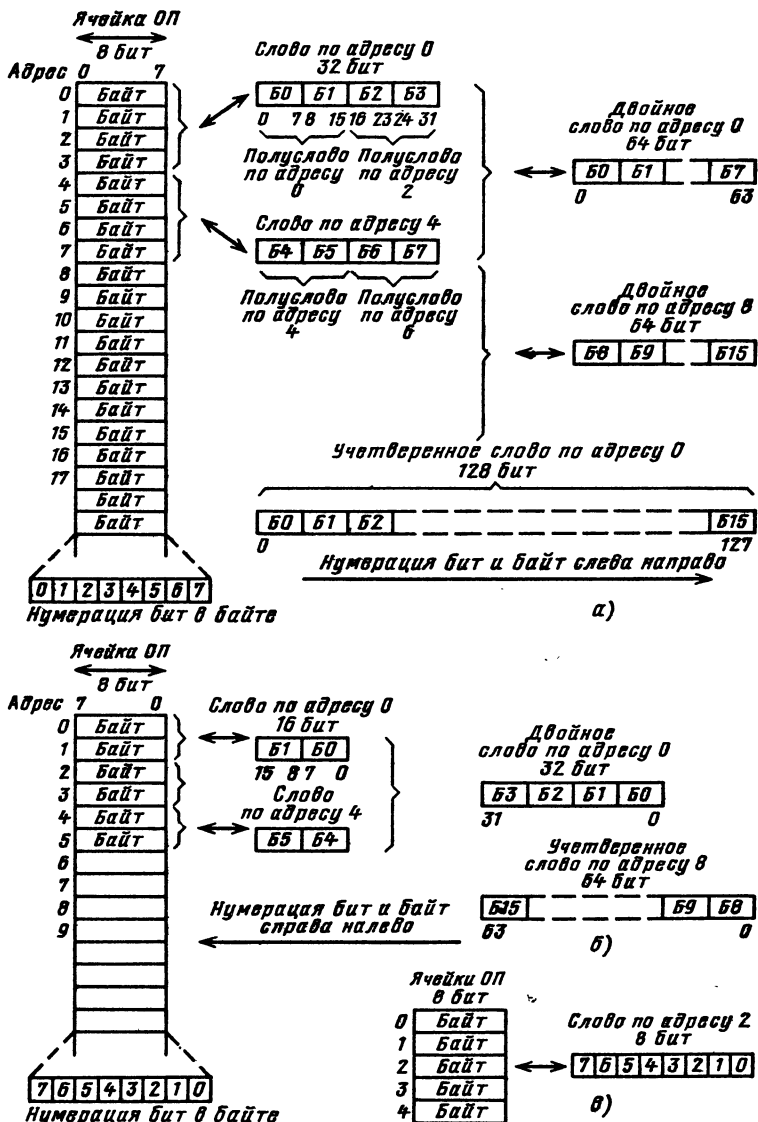


Рис. 9.2. Адресные структуры основных (оперативных) памяти в ЕС ЭВМ (а), в малых ЭВМ (СМ-1420 и др.) (б) и в микропроцессорах (К580) (в)

в неявной форме кодом операции команды, или непосредственно указывается в коде команды.

Адрес в ЕС ЭВМ представляет собой целое 24-разрядное двоичное число, позволяющее адресовать до 16 777 216 байт (16 Мбайт). В ЕС ЭВМ используются следующие форматы информации фиксированной длины: 1) байт; 2) полуслово (два последовательно расположенных в памяти байта); 3) слово (4 байта); 4) двойное слово (8 байт); 5) учетверенное слово (16 байт). Кроме того, используются слова (или поля) переменной длины для представления десятичных чисел (в формате от 1 до 16 байт), логических кодов и алфавитно-цифровых данных (в формате от 1 до 256 байт).

Адрес слова переменной длины (адрес его левого байта) может быть любым в пределах адресной сетки машины, что обеспечивает размещение в памяти полей переменной длины без просветов.

Для упрощения и убыстрения процедур выполнения команд в ряде ЭВМ на адресацию единиц информации фиксированной длины накладываются определенные ограничения (так называемые целочисленные границы). Например, в ЕС ЭВМ II очереди адреса полуслов, слов, двойных и учетверенных слов рекомендуется назначать кратными соответственно 2, 4, 8 и 16, при этом двоичные адреса указанных единиц информации содержит 0 соответственно в одном, двух, трех и четырех младших разрядах. Адрес команды может быть только четным.

#### **9.4. Проблема выбора структуры и формата команд. Кодирование команд**

Все возможные преобразования дискретной информации могут быть сведены к четырем основным видам: 1) передача информации в пространстве (например, из одного блока ЭВМ в другой); 2) передача информации во времени (хранение); 3) логические (поразрядные) операции; 4) арифметические операции. ЭВМ, являющаяся универсальным преобразователем дискретной информации, выполняет все указанные виды преобразований.

Обработка информации (решение задач) в ЭВМ осуществляется автоматически путем программного управления. Программа представляет собой алгоритм обработки информации (например, решения математической задачи), записанный в виде последовательности команд, которые должны быть выполнены машиной для получения решения задачи.

*Команда* представляет собой код, определяющий операцию вычислительной машины и данные, участвующие в операции.

Команда содержит также в явной или неявной форме информацию об адресе, по которому помещается результат операции, и об адресе следующей команды.

По характеру выполняемых операций различают следующие основные группы команд: а) команды арифметических операций для чисел с фиксированной и плавающей точками; б) команды десятичной арифметики; в) команды логических (поразрядных) операций (И, ИЛИ и др.); г) команды передачи кодов; д) команды операций ввода-вывода; е) команды управления порядком исполнения команд (команды передачи управления); ж) команды задания режима работы машины и др.

В команде, как правило, содержатся не сами операнды, а информация об адресах ячеек памяти или регистрах, в которых они находятся.

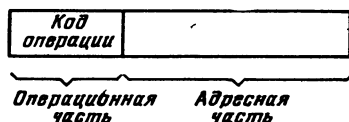
Код команды можно представить состоящим из нескольких частей или полей, имеющих определенное функциональное назначение при кодировании командной информации. Команда в общем случае состоит из операционной и адресной частей (рис. 9.3). В свою очередь, эти части, что особенно характерно для адресной части, могут состоять из нескольких полей.

*Операционная часть* содержит *код операции* (КОп), который задает вид операции (сложение, умножение, передача и т. д.). *Адресная часть команды* содержит информацию об адресах операндов и результата операции, а в некоторых случаях информацию об адресе следующей команды.

*Структура команды* определяется составом, назначением и расположением полей в команде. *Форматом команды* называют ее структуру с разметкой номеров разрядов (бит), определяющих границы отдельных полей команды, или с указанием числа бит в определенных полях.

Важной и сложной проблемой при проектировании ЭВМ является выбор структуры и формата команды, т. е. ее длины, назначения и длины отдельных ее полей. Естественно стремление разместить в команде в возможно более полной форме информацию о предписываемой командой операции. Однако в условиях, когда в современных ЭВМ значительно возросло число выполняемых различных операций и соответственно команд (система команд ЕС ЭВМ содержит около 200 команд) и значительно увеличилась емкость адресуемой основной памяти (в ЕС ЭВМ

Рис. 9.3. Общая структура команд



до 16 Мбайт), это приводит к недопустимо большой длине формата команды.

Действительно, число двоичных разрядов, отводимых под код операции, должно быть таким, чтобы можно было представить все выполняемые машинные операции. Если ЭВМ выполняет  $M$  различных операций, то число разрядов в коде операции

$$n_{к.о} \geq \log_2 M. \quad (9.1)$$

Например, при  $M=200$   $n_{к.о}=8$ .

Если основная память содержит  $S$  адресуемых ячеек (байт), то для представления только одного адреса (например, адреса первого операнда) необходимо в команде иметь адресное поле для одного операнда с числом разрядов

$$n_A \geq \log_2 S. \quad (9.2)$$

Например, при  $S=16$  Мбайт  $n_A=24$ .

Вместе с тем для упрощения аппаратуры и повышения быстродействия ЭВМ длина формата команды должна быть согласована с выбираемой исходя из требований к точности вычислений длиной обрабатываемых машиной слов (операндов), составляющей для большинства применений 16—32 бит, с тем чтобы для операндов и команд можно было эффективно использовать одни и те же память и аппаратурные средства обработки информации. Формат команды должен быть по возможности короче, укладываться в машинное слово или полуслово, а для ЭВМ с коротким словом (8—16 бит) должен быть малократным машинному слову. Решение проблемы выбора формата команды значительно усложняется в малых и микроЭВМ и микропроцессорах, работающих с коротким словом.

Отмечавшиеся ранее характерные для процесса развития ЭВМ расширение системы (набора) команд и увеличение емкости основной памяти, а особенно создание малых и микроЭВМ с коротким словом, потребовали разработки методов укорочения формата команды. При решении этой проблемы существенно видоизменилась структура команды, получили развитие различные способы адресации информации.

Проследим изменения классических структур команд.

Чтобы команда содержала в явном виде всю необходимую информацию о задаваемой операции, она должна, как это показано на рис. 9.4, а, содержать поле кода операции и четыре адреса для указания ячеек памяти, содержащих два операнда, участвующих в операции, ячейки, в которую помещается результат операции, и ячейки, содержащей следующую команду. Такой порядок выборки команд называется *принудительным*. Он ис-

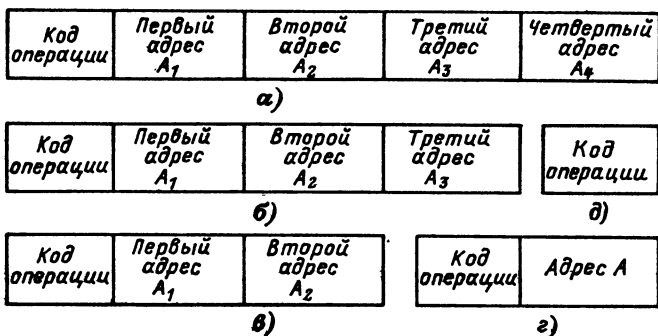


Рис. 9.4. Структуры команд:

*а* — четырехдресная; *б* — трехдресная; *в* — двухдресная; *г* — однодресная; *д* — безадресная

пользовался в некоторых первых моделях ЭВМ. Для рассмотренного выше примера с  $M=200$  и  $S=16$  Мбайт длина четырех-адресной команды составила бы 104 бит. Такой формат команды окажется трудно реализуемым и неэффективным. Четырех-адресные команды в настоящее время не применяются.

Можно установить, как это принято для большинства машин, что после выполнения данной команды, расположенной по адресу  $K$  (и занимающей  $L$  ячеек), выполняется команда из  $(K+L)$ -й ячейки. Такой порядок выборки команд называется *естественным*. Он нарушается только специальными командами. В таком случае отпадает необходимость указывать в команде в явной форме адрес следующей команды. В трехдресной команде (рис. 9.4, б) первый и второй адреса указывают ячейки памяти, в которых расположены операнды, а третий определяет ячейку, в которую помещается результат операции. Операция, описываемая трехдресной командой, может быть символически представлена в виде

$$ОП [A_3] := ОП [A_1] * ОП [A_2],$$

где знак  $*$  заменяет символ операции ( $+$ ,  $-$  и т. п.).

Можно условиться, что результат операции всегда помещается на месте одного из операндов, например первого. Получим двухдресную команду (рис. 9.3, в)

$$ОП [A_1] := ОП [A_1] * ОП [A_2],$$

т. е. для результата используется подразумеваемый адрес.



В одноадресной команде (рис. 9.4, *з*) подразумеваемые адреса имеют уже и результат операций, и один из операндов. Один из операндов указывается адресом в команде, в качестве второго используется содержимое регистра процессора, называемого в этом случае регистром результата или аккумулятором (Акк). Результат операции записывается в тот же регистр:

$$Акк := Акк * ОП [A].$$

Наконец, в некоторых случаях возможно использование безадресных команд (рис. 9.4, *д*), когда подразумеваются адреса обоих операндов и результата операции, например при работе со стековой памятью.

С точки зрения программиста наиболее естественны и удобны трехадресные команды. Однако из-за необходимости иметь большее число разрядов для представления адресов и кода операции длина трехадресной команды становится недопустимо большой, и ее не удастся разместить в машинном слове. Следует отметить, что очень часто в качестве операндов используются результаты предыдущих операций, хранимые в регистрах машины. В этом случае выполняемая операция приобретает характер одно- или двухадресный, при этом трехадресный формат используется неэффективно. По указанным причинам в современных ЭВМ применяют, как правило, двух- и одноадресные команды и их модификации.

*Способ расширения кодов операций.* В машинах с коротким словом практически невозможно в одном формате команды, т. е. при фиксированном назначении ее полей, кодировать большое число различных операций и одновременно иметь гибкую форму адресации операндов. Это противоречие в машинах с коротким словом преодолевается расширением кодов операций в команде. Для задания небольшой группы основных операций (арифметических и др.) используется короткий код операции, а получаемая при этом сравнительно большая адресная часть команды позволяет реализовать гибкую, например двухадресную с многими модификациями, адресацию. Для задания других операций используются более длинные (*расширяемые*) коды операций, при этом сокращаемая адресная часть оставляет возможность лишь для более простой, например одноадресной, адресации операндов. В пределе расширяемый код операций занимает весь формат команды (безадресная команда).

Обычно в ЭВМ используется несколько форматов команд разной длины.

Приведенные на рис. 9.4 форматы команд достаточно схематичны. В действительности адресные поля команд большей

частью содержат не сами адреса, а только информацию, позволяющую определить действительные (исполнительные) адреса операндов в соответствии с используемыми в командах способами адресации.

## 9.5. Способы адресации

Следует различать понятия *адресный код* в команде  $A_k$  и *исполнительный адрес*  $A_{\text{и}}$ . Адресный код — это информация об адресе операнда, содержащаяся в команде. Исполнительный адрес — это номер ячейки памяти, к которой производится фактическое обращение. В современных ЭВМ адресный код, как правило, не совпадает с исполнительным адресом.

Выбор способов адресации, формирования исполнительного адреса и преобразования адресов является одним из важнейших вопросов разработки ЭВМ. Рассмотрим способы адресации, используемые в современных ЭВМ.

*Подразумеваемый операнд.* В команде не содержится явных указаний об адресе операнда; операнд подразумевается и фактически задается кодом операции команды. Данный способ используется не часто, однако имеется несколько важных случаев его применения. В качестве примера можно привести команды подсчета, в которых к некоторому числу (содержимому счетчика) прибавляется фиксированное приращение, чаще единица младшего разряда. Один из операндов — число в счетчике — обычно адресуется явным методом, второй операнд — приращение — не адресуется, в памяти машины не содержится и является подразумеваемым.

*Подразумеваемый адрес.* В команде не содержится явных указаний об адресе участвующего в операции операнда или адреса, по которому помещается результат операции, но этот адрес подразумевается. Например, команда может содержать адреса обоих операндов, участвующих в операции, при этом подразумевается, что результат операции помещается по адресу одного из операндов, или команда указывает только адрес одного операнда, а адрес второго, которым является содержимое специального регистра (называемого регистром результата или аккумулятором), подразумевается.

*Непосредственная адресация.* В команде содержится не адрес операнда, а непосредственно сам операнд. При непосредственной адресации не требуется обращения к памяти для выборки операнда и ячейки для его хранения. Это способствует уменьшению времени выполнения программы и занимаемого ею объема памяти. Непосредственная адресация удобна для хранения различного рода констант; однако следует иметь в виду, что

при этом способе адресации длина операнда короче кода команды, поскольку часть разрядов команды занята под код операции.

**Прямая адресация.** Исполнительный адрес совпадает с адресной частью команды. Этот способ адресации был общепринятым в первых вычислительных машинах и продолжает применяться в настоящее время в комбинации с другими способами.

В указанной форме непосредственная адресация реализуется в ЭВМ со сравнительно длинным машинным словом (32 разряда и более), например в ЕС ЭВМ. Особенности непосредственной адресации в малых ЭВМ и микропроцессорах изложены в § 9.1 и 10.2.

**Относительная адресация или базирование.** Исполнительный адрес определяется суммой адресного кода команды  $A_K$  и некоторого числа  $A_6$ , называемого **базовым адресом**:

$$A_n = A_6 + A_K.$$

Для хранения базовых адресов в машине могут быть предусмотрены регистры или специально выделенные для этой цели ячейки памяти (**базовые регистры**). В команде выделяется поле  $B$  для указания номера базового регистра.

Относительная адресация позволяет при меньшей длине адресного кода команды обеспечить доступ к любой ячейке памяти. Для этого число разрядов в базовом адресе выбирают таким, чтобы можно было адресовать любую ячейку ОП, а адресный код  $A_K$  самой команды используют для представления лишь сравнительно короткого «смещения» (обозначают буквой  $D$ ). Смещение  $D$  определяет положение операнда относительно начала массива, задаваемого базовым адресом  $A_6$ . Рисунок 9.5 поясняет процесс формирования исполнительного адреса.

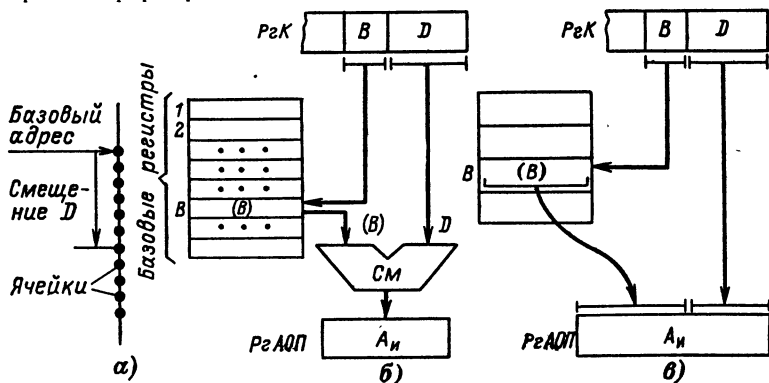


Рис. 9.5. Базирование (относительная адресация):

а — образование адреса элемента одномерного массива; б — формирование исполнительного адреса суммированием; в — формирование исполнительного адреса совмещением

Большей частью исполнительный адрес при базировании образуется с помощью сумматора согласно выражению (рис. 9.5)

$$A_n = \begin{cases} (B) + D, & \text{если } B \neq 0; \\ D, & \text{если } B = 0, \end{cases}$$

где  $B$  и  $D$  — коды (числа), стоящие в соответствующих полях команды;  $(B)$  — содержимое регистра с номером  $B$ . При  $B = 0$  — относительная адресация блокируется.

Более подробно обращение к ОП при относительной адресации можно представить в следующем виде:

**если**  $PzK[B] = 0$  **то**  $PzAOP := PzK[D]$

**иначе**  $PzAOP := PzB + RgK[D]$ ;

*Счит:*  $PzИОП := ОП[PzAOP]$ ;

Суммирование при образовании  $A_n$  связано с потерей времени. Поэтому применяют также формирование исполнительного адреса методом совмещения (рис. 9.5, в). В этом случае базовый адрес содержит старшие, а смещение — младшие разряды исполнительного адреса, которые объединяются в  $PzAOP$  согласно операции конкатенации слова:

$$PzAOP := PzB | PzK[D];$$

Однако при совмещении базовый адрес может задавать не любую ячейку, а только те ячейки, адреса которых содержат 0 в младших разрядах, соответствующих смещению.

Относительная адресация обеспечивает так называемую перемещаемость программ, т. е. возможность передвижения программ в памяти без изменений внутри самой программы.

**Укороченная адресация.** Для уменьшения длины кода команды часто применяется так называемая укороченная адресация. Суть ее сводится к тому, что в команде задаются только младшие разряды адресов, старшие разряды при этом подразумеваются нулевыми. Такая адресация позволяет использовать только небольшую группу фиксированных ячеек с начальными (короткими) адресами и поэтому может применяться лишь совместно с другими способами адресации.

**Регистровая адресация** есть частный случай укороченной, когда в качестве фиксированных ячеек с короткими адресами используются регистры (ячейки сверхоперативной или местной памяти) процессора. Например, если таких регистров 16, то для адреса достаточно четырех двоичных разрядов. Регистровая адресация наряду с сокращением длины адресов операндов позволяет увеличить скорость выполнения операций, так как уменьшается число обращений к ОП.

**Косвенная адресация.** Адресный код команды указывает адрес ячейки памяти, в которой находится адрес операнда или команды. Таким образом, косвенная адресация может быть иначе определена как «адресация адреса».

На косвенную адресацию указывает код операции команды, а в некоторых ЭВМ в команде отводится специальный разряд (указатель адресации — УА), и цифра 0 или 1 в нем указывает, является адресная часть команды прямым адресом или косвенным. Обращение к ОП за операндом при косвенной адресации представляет собой следующую процедуру:

$R_2AOP : = R_2K [A];$   
 Счит:  $R_2ИОП : = ОП [R_2AOP];$   
 если  $УА=0$  то идти к М иначе  $R_2AOP : = R_2ИОП;$   
 Счит:  $R_2ИОП : = ОП [R_2AOP];$   
 М:  $R_2АЛУ : = R_2ИОП;$

В некоторых ЭВМ используется многоступенчатая косвенная адресация. В этом случае ячейки памяти содержат также разряд — указатель косвенной адресации (УА). Если этот разряд указывает на продолжение косвенной адресации, то машина последовательно выбирает из памяти адреса до тех пор, пока не будет найдена ячейка, в которой разряд-указатель определит прямую адресацию. Адрес из этой последней ячейки и является искомым исполнительным адресом.

*Косвенная адресация широко используется в малых и микроЭВМ, имеющих короткое машинное слово, для преодоления ограничений короткого формата команды.*

Рассмотрим широко применяемое в микропроцессорах, малых и микроЭВМ совместное использование укороченной (регистровой) и косвенной адресаций (рис. 9.6). Пусть необходимо передать число 4527 из  $R_25$  в ячейку ОП 1765. Длина адресных

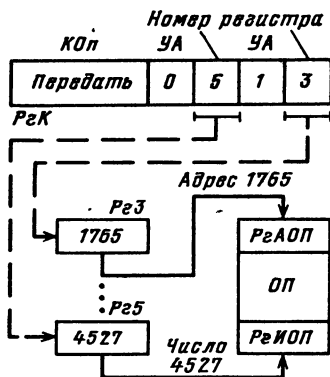


Рис. 9.6. Совместное использование регистровой прямой и регистровой косвенной адресаций для преодоления ограничений короткого слова команды

полей команды достаточна лишь для указания коротких номеров регистров, она не позволяет задать в команде полный адрес ячейки памяти. Поэтому операнд 4527 указывается регистровой прямой адресацией ( $UA=0$ ), а для задания адреса 1765 приходится воспользоваться регистровой косвенной адресацией ( $UA=1$ ), при которой в команде указывается номер регистра (в рассматриваемом примере  $R_2$ ), предварительно загруженного полным адресом ячейки, в которую производится передача.

*Автоинкрементная и автодекрементная адресации.* Поскольку регистровая косвенная адресация требует предварительной загрузки регистра из ОП косвенным адресом, что связано с потерей времени, такой тип адресации особенно эффективен при обработке массива данных, если имеется механизм автоматического приращения или уменьшения содержимого регистра при каждом обращении к нему, называемый соответственно автоинкрементной и автодекрементной адресацией. В этом случае достаточно 1 раз загрузить в регистр адрес первого обрабатываемого элемента массива, а затем при каждом обращении к регистру в нем в результате инкрементной (декрементной) процедуры формируется адрес следующего элемента массива.

При *автоинкрементной адресации* по содержимому регистра сначала содержимое регистра используется как адрес операнда, а затем получает приращение, равное числу байт в элементе массива. При *автодекрементной адресации* сначала содержимое указанного в команде регистра уменьшается на число, равное числу байт в элементе массива, а затем используется как адрес операнда.

Автоинкрементная и автодекрементная адресации могут рассматриваться как упрощенный вариант индексации — весьма важного механизма преобразования адресных частей команд (см. § 9.9) и организации вычислительных циклов, поэтому их часто называют *автоиндексацией*.

*Адресация слов переменной длины.* Эффективность вычислительных систем, предназначенных для обработки данных (экономических, плановых и др.), повышается, если имеется возможность выполнять операции со словами переменной длины. В этом случае в машине должна быть предусмотрена адресация слов переменной длины, которая обычно реализуется путем указания в команде местоположения в памяти начала слова и его длины.

Обычно в ЭВМ одновременно используется несколько типов адресации. Тип адресации указывается либо неявно кодом операции, либо в явной форме в специальном поле адресной части команды.

## 9.6. Стековая адресация

*Стековая память* (см. гл. 4), реализующая безадресное задание операндов, является эффективным элементом современной архитектуры ЭВМ, особенно широко используемым в микропроцессорах, малых и микроЭВМ, а также в некоторых суперЭВМ. Учитывая своеобразие стековой адресации, ее рассмотрение выделено в отдельный параграф.

Стек представляет собой группу последовательно пронумерованных регистров (*аппаратурный стек*) или ячеек памяти, снабженных указателем стека (обычно регистром) (УС), в котором автоматически при записи и считывании устанавливается номер (адрес) последней занятой ячейки стека (*вершины стека*). При операции записи заносимое в стек слово помещается в следующую по порядку свободную ячейку стека, а при считывании из стека извлекается последнее поступившее в него слово. Таким образом, в стеке реализуется правило *«последний пришел — первый ушел»*.

Указанное правило при обращении к стеку реализуется автоматически, и поэтому при операциях со стеком возможно безадресное задание операнда — команда не содержит адреса ячейки стека, но содержит адрес (или он подразумевается) ячейки памяти или регистра, откуда слово передается в стек или куда помещается из стека.

Механизм стековой адресации поясняется на рис. 9.7. При выполнении команды передачи в стек слова из регистра или ячейки ОП сначала указатель стека увеличивается на 1 (*в перевернутом стеке* уменьшается на 1), а затем слово помещается в ячейку стека, указываемую УС. При команде загрузки из стека регистра или ячейки памяти сначала слово извлекается из вершины стека, а затем указатель стека уменьшается на 1 (*в перевернутом стеке* увеличивается на 1)<sup>1</sup>. Как это ни кажется на первый взгляд удивительным, но при соответствующем расположении операндов в стеке можно вычислять выражения полностью безадресными командами, указывающими только вид операции. Такая команда извлекает из стека в соответствии с кодом операции один или два операнда, выполняет над ними предписанную операцию и заносит результат в стек.

Вычисления с использованием стековой памяти удобно описывать и программировать с помощью польской инверсной (бесскобочной) записи арифметических выражений ПОЛИЗ. Эта запись производится по следующему правилу: читаем арифмети-

---

<sup>1</sup> В случае использования памяти с побайтной адресацией при занесении слова в стек и извлечении слова из стека содержимое УС изменяется на величину  $L$ , где  $L$  — количество байт в слове.

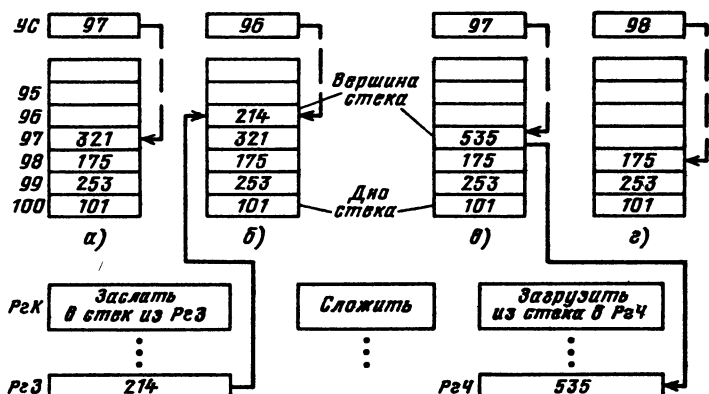


Рис. 9.7. Стекковая адресация в «перевернутом стеке»:

а — исходное состояние стека; б — стек после выполнения команды «Заслать в стек из Рег3»; в — стек после выполнения команды «Сложить»; г — стек после выполнения команды «Загрузить из стека в Рег4»

ческое выражение слева направо и последовательно друг за другом выписываем встречающиеся операнды. Как только окажется, что все операнды некоторой операции выписаны, записываем знак этой операции и продолжаем выписывать операнды. Если операция имеет операндом результат некоторой предыдущей операции и знак последней выписан, то считаем этот операнд выписанным.

Например, выражение

$$(k + l - m)(p - s) \quad (9.3)$$

в ПОЛИЗ имеет вид

$$k \ l \ + \ m \ - \ p \ s \ - \ \times. \quad (9.4)$$

Выражение в ПОЛИЗ не содержит скобок, но порядок действий определяет однозначно. При использовании стековой памяти последовательность символов в выражении ПОЛИЗ, например (9.4), может рассматриваться как программа вычисления исходного арифметического выражения (рис. 9.8), если под буквами понимать команды засылки, содержащие только адреса в ОП соответствующих операндов, засылаемых в стек, а под знаками операций — *безадресные команды*, содержащие только коды операций. Команда второго типа

Адрес k
Адрес l
+
Адрес m
-
Адрес p
Адрес s
-
×

Рис. 9.8. Программа вычисления выражения (9.4) с использованием стековой памяти



инициирует извлечение из стека двух (или одного) слов, выполнение над ними указанной в команде операции и засылку результата в вершину стека.

Безадресные команды на основе стековой адресации предельно сокращают формат команд, экономят память и способствуют повышению производительности ЭВМ.

Однако при такой структуре команд возникают осложнения с построением команд передачи управления и работы с периферийными устройствами.

В современной архитектуре процессоров и микропроцессоров стек и стековая адресация широко используются при организации переходов к подпрограммам и возврате от них, а также в системах прерывания. Весьма широко стеки и безадресные команды используются в вычислительном комплексе «Эльбрус» (см. гл. 15).

## 9.7. Команды, процедуры и микропрограммы передачи управления в программах

Рассмотрим команды, управляющие порядком исполнения команд.

При естественном порядке после выполнения очередной команды выбирается команда, расположенная в следующей по порядку ячейке памяти. Обычно адрес команд хранится в специальном регистре, называемом счетчиком команды (СчК), содержимое которого после выполнения каждой команды автоматически увеличивается на 1, а если память имеет побайтную адресацию, то оно увеличивается на столько, сколько байт содержит текущая команда (*приращение адреса команды  $L_K$* ).

Выборка очередной и формирование адреса следующей команды (АСК), часто называемого «продвинутым адресом», происходит (при естественном порядке) следующим образом (рис. 9.9):

$$\begin{aligned} P_2 A O П &:= СчК; \\ Счит : P_2 И O П &:= O П [P_2 A O П]; \\ P_2 К &:= P_2 И O П; \end{aligned}$$

. . . . .

(Выполнение текущей команды)

. . . . .

$$A C K_{E П} : C ч К := C ч К + L_K;$$

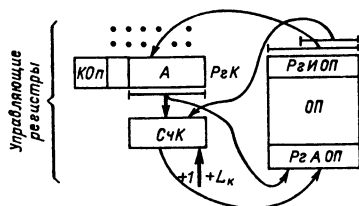


Рис. 9.9. Формирование адреса следующей команды при естественном порядке выборки команд и при его нарушении командами перехода и командой «Выполнение»

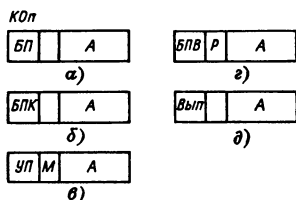


Рис. 9.10. Упрощенные структуры команд перехода безусловно (а), безусловного по косвенному адресу (б), условного (в), безусловного с возвратом (г) и команды «Выполнение» (д)

Естественный порядок выполнения команд может быть нарушен: 1) командами перехода (командами передачи управления); 2) командами замещения (*Выполнение*); 3) сменой состояния программы; 4) запросами прерывания программы.

*Команды перехода (передачи управления)*. Известны многочисленные варианты команд перехода, однако общий принцип состоит в том, что адресная часть команды перехода непосредственно или после суммирования с содержимым базового регистра передается в счетчик команд, и, следовательно, после данной команды может быть выполнена команда из произвольной ячейки памяти, номер которой определяется адресной частью команды перехода.

Для упрощения изложения материала рассмотрим процедуру выполнения команд переходов без относительной адресации.

Различают два основных вида команд перехода: *безусловный* переход (БП) и *условный* (УП). Упрощенные структуры этих команд показаны на рис. 9.10.

*Команда БП* (рис. 9.10, а) совершает переход всегда независимо от каких-либо условий. Обыкновенно следующей за командой БП выполняется команда, адрес которой указан в адресной части команды БП, т. е.

$$АСК_{БП} : СчК : = РзК [A];$$

Другим вариантом команды БП является *безусловный переход по косвенному адресу* (БПК) (рис. 9.10, б). На косвенную адресацию указывает код операции команды или специальное поле в формате команды, определяющее тип адресации. Управление передается команде, расположенной в ячейке, адрес которой указан в адресной части команды БПК. В этом случае вы-

полняется следующая процедура:

$$\begin{aligned} P_2 A O П : &= P_2 K [A]; \\ Счит : P_2 И O П : &= O П [P_2 A O П]; \\ A C K_{Б П К} : C ч К : &= P_2 И O П [A]; \end{aligned}$$

При *условном переходе* адрес следующей команды зависит от выполнения некоторого условия. Обычно, если условие выполняется, происходит переход к команде по адресу, указанному в адресной части команды УП. В противном случае выбирается следующая по порядку команда, адрес которой определяется, как обычно, содержимым *СчК*, увеличенным на *приращение адреса команды*  $L_K$ . Так же как и при БП, в команде УП могут использоваться относительная и косвенная адресации.

Условия перехода (например, результат предыдущей команды равен, больше или меньше 0, переполнение разрядной сетки и др.) задаются кодом операции УП или в виде отдельного поля *маска условия*  $M$  в команде УП (рис. 9.10, в). В этом случае следующая команда определяется по правилу

$$\begin{aligned} \text{если } Ус [M] = 1 \text{ то } A C K_{У П} : C ч К : &= P_2 K [A] \text{ иначе} \\ A C K_{Е П} : C ч К : &= C ч К + L_K; \end{aligned}$$

Здесь  $Ус [M]$  — логическая функция (условие), заданная (заданное) маской  $M$ . Если условие перехода задается кодом операции команды, то в приведенном выражении  $Ус [M]$  надо заменить на  $Ус [K O n]$ .

Обычно выполнение машинной команды сопровождается выработкой кода признака результата  $ПР$ , формируемого в специальном регистре  $P_2 ПР$ . Команда УП анализирует сформированный предыдущей командой  $ПР$ . Смысл кодов  $ПР$  может быть установлен различным для разных операций. Например, в ЕС ЭВМ двухразрядный признак результата  $ПР$  при выполнении арифметических операций принимает значения, приведенные в табл. 9.1 ( $ПР [0]$  и  $ПР [1]$  — нулевой и первый разряды  $ПР$ ).

С помощью двухразрядной маски можно задать в качестве условия УП любое значение  $ПР$ .

Т а б л и ц а 9.1

Результат операции	Код $ПР$		Маска условия (ЕС ЭВМ)			
	$ПР [0]$	$ПР [1]$	$M [0]$	$M [1]$	$M [2]$	$M [3]$
Равен 0	0	0	1	0	0	0
Меньше 0	0	1	0	1	0	0
Больше 0	1	0	0	0	1	0
Переполнение	1	1	0	0	0	1

В машинах ЕС ЭВМ используется четырехразрядная маска, и это дает возможность задавать в качестве условия перехода выполнение не только любого из  $PR$ , но и их дизъюнкции (если в маске  $M$  несколько 1).

В этом случае имеем

$$Ус [M] = M [0] \overline{PR [0]} \overline{PR [1]} \vee M [1] \overline{PR [0]} PR [1] \vee \\ \vee M [2] PR [0] \overline{PR [1]} \vee M [3] PR [0] PR [1].$$

Отметим, что при  $M = 1111$  команда УП выполняет безусловный переход, а при  $M = 0000$  (пустая команда) действует естественный порядок выборки команд. Таким образом, исключается необходимость в отдельной команде безусловного перехода.

Команды условных переходов позволяют реализовать программы с разветвлениями в зависимости от промежуточных результатов вычислений или состояния машины.

Важным случаем передачи управления являются *безусловные переходы к подпрограммам*. Их особенность состоит в том, что помимо перехода они должны обеспечить по окончании подпрограммы возврат к исходной программе, к той точке ее, откуда был совершен переход. Обычно для переходов к подпрограммам используется специальная команда *Безусловный переход с возвратом* (БПВ). По этой команде (рис. 9.11) сначала адрес возврата  $A_{\text{воз}}$ , т. е. содержимое  $CчК$  (увеличенное на «приращение адреса команды»<sup>1</sup>  $L_K$ ), запоминается по адресу  $P$ , указанному в команде БПВ<sup>2</sup>, затем в счетчик команд заносится содержимое поля  $A$  команды БПВ, т. е. адрес  $A$  начала подпрограммы. В конце подпрограммы размещается команда возврата, которая представляет собой команду БПК, указывающую путем косвенной адресации адрес ячейки (или регистра), в которой находится адрес  $A_{\text{воз}}$ .

Формально всю эту процедуру после приема в  $PзК$  команды БПВ можно представить в следующем виде (см. рис. 9.9 и 9.11):

$$CчК := CчК + L_K \text{ (образование адреса возврата } A_{\text{воз}});$$

$$PзИОП := CчК;$$

$$PзАОП := PзК [P];$$

Запись:  $ОП [PзАОП] := PзИОП$  (запоминание адреса возврата  $A_{\text{воз}}$  в ячейке  $P$ );

$$CчК := PзК [A] \text{ (передача в } CчК \text{ адреса начала подпрограммы);}$$

<sup>1</sup> Для ЭВМ с пословной адресацией и длиной команды, равной слову,  $L_K = 1$ .

<sup>2</sup> Обычно это один из общих регистров.

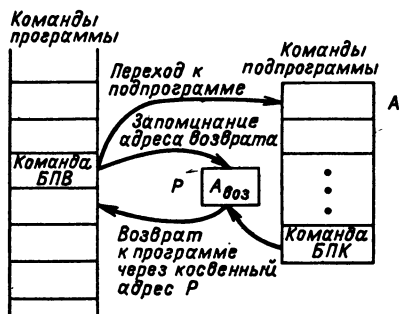


Рис. 9.11. Обращение к подпрограмме с помощью команды безусловного перехода с возвратом

(Выполнение подпрограммы)

(Считывание и передача в  $PgK$  заключающей подпрограмму команды БПК, содержащей в поле  $A$  косвенный адрес  $P$ )

$PgAOP := PgK[A];$

Счит:  $PgIOП := ОП[PgAOP]$  (извлечение адреса возврата  $A_{воз}$ );

$АСКБПВ: СчК := PgIOП[A]$  (возврат к основной программе — команде в ячейке  $A_{воз}$ );

Операция замещения, реализуемая командой *Выполнение (Вып)*, состоит в том, что вместо очередной команды, соответствующей естественному порядку выборки команд, исполняется замещающая команда, указываемая адресной частью команды *Вып*, а затем, если только замещающая команда не оказалась командой перехода, восстанавливается приостановленный на время выполнения команды *Вып* естественный порядок выборки команд. Команда *Вып* должна сохранять неизменным содержимое  $СчК$ . Поэтому адрес замещающей команды берется не из  $СчК$ , а из  $PgK$  (см. рис. 9.10). Соответствующую процедуру после приема в  $PgK$  команды *Вып* можно представить в виде

$PgAOP := PgK[A];$

Счит:  $PgIOП := ОП[PgAOP];$

$PgK := PgIOП$  (выборка замещающей команды);

.....  
Выполнение замещающей команды

.....  
 $СчК := СчК + L_K$  (восстановление естественного порядка выборки команд);

## 9.8. Индексация

Характерным для реализуемых на ЭВМ методов решения математических задач и обработки данных является циклич-

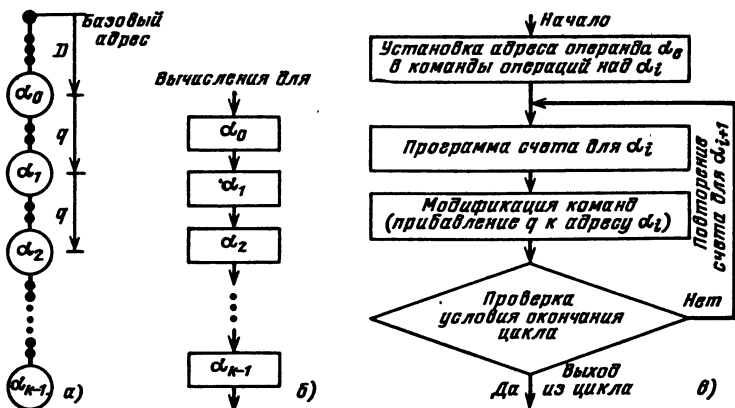


Рис. 9.12. Вычислительный цикл и автоматическая модификация адресов команд

ность вычислительных процессов, при которой повторяются одни и те же процедуры, но над различными операндами (элементами информационных массивов), расположенными упорядоченно в памяти. Поскольку операнды, обрабатываемые при повторениях цикла, имеют разные адреса, можно для каждого повторения составить свою последовательность команд, отличающихся адресными частями. На рис. 9.12, а в качестве простого примера показан одномерный информационный массив, в котором выделены кружками подлежащие обработке операнды  $\alpha_0, \alpha_1, \dots, \alpha_{k-1}$ , расположенные последовательно со сдвигом на  $q$  ячеек. Программа состоит из групп команд (рис. 9.12, б), повторяющих один и тот же цикл вычислений, причем каждая последующая отличается от предыдущей только приращением на  $q$  адреса операнда  $\alpha_i$ . Однако при таком подходе программа решения задачи оказывается слишком длинной, а ее составление — чрезмерно трудоемким.

Программирование вычислительных циклов существенно упростится, если после каждого выполнения цикла обеспечить автоматическое изменение в соответствующих командах их адресных частей согласно расположению в ОП обрабатываемых операндов. Такой процесс называется *модификацией команд*, точнее, адресных частей команд. Модификация команд основана на возможности выполнения над кодами команд или их частями арифметических и логических операций. В рассматриваемом примере модификация команд, обрабатывающих операнд  $\alpha_i$ , состоит в увеличении их адресной части на  $q$  после каждого цикла счета.

Благодаря автоматической модификации команд длина программы существенно сокращается, так как она содержит лишь группу команд для одного прохождения цикла и команды служебных операций, связанных с модификацией команд и управлением вычислительным циклом. Управление вычислительным циклом должно обеспечить повторение цикла нужное число раз, а затем выход из него. Сказанное поясняется схемой на рис. 9.12, в.

Контроль окончания циклических вычислений и выход из цикла могут производиться по числу проделанных повторений цикла, т. е. по содержанию некоторого счетчика, изменяемого на единицу при каждом повторении цикла, или путем задания некоторого предела для изменения адресной части (адресного кода) в модифицируемой команде.

Автоматическая модификация команд и управление вычислительными циклами в современных ЭВМ обеспечиваются механизмом *индексации*. Это понятие включает в себя специальный способ кодирования команд, командные и аппаратурные средства задания и выполнения модификации команд и управления вычислительными циклами. Упомянутые средства часто называют *индексной арифметикой*.

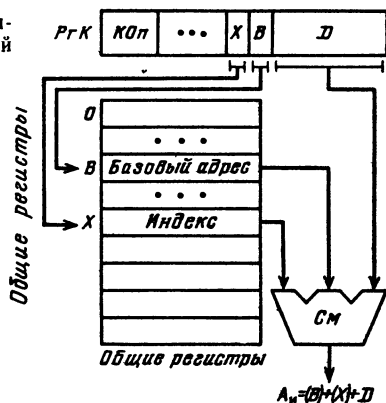
Как способ модификации команд индексация является развитием метода базирования (относительной адресации), с которым она часто используется совместно. Как средство управления вычислительными циклами индексация основана на использовании специализированных команд условного перехода.

Для выполнения индексации в машину вводятся так называемые *индексные регистры*<sup>1</sup>, в качестве которых используются ячейки сверхоперативной (местной) памяти или триггерные регистры. В формате команды выделяется поле  $X$  для указания номера индексного регистра. Если  $(X)=0$ , то это служит указанием, что данная команда не модифицируется. Исполнительный адрес при индексации формируется путем сложения адресного кода команды (смещения) с содержимым индексного регистра (*индексом*), а при наличии базирования — и с базовым адресом. Именно этот характерный случай одновременного использования базирования и индексации показан на рис. 9.13. В схеме имеются *общие регистры*, которые программист может использовать в качестве базовых, индексных и для других целей. Пусть  $B$  и  $X$  — номера регистров, хранящих соответственно базовый адрес ( $B$ ) и индекс ( $X$ ).

---

<sup>1</sup> Наименование индексных регистров связано с тем, что первоначально они предназначались для хранения индекса текущего элемента массива  $a_i$  ( $i=0, 1, 2, \dots$ ). Хотя назначение индексных регистров по сравнению с первоначальным значительно расширилось, термин сохранился.

Рис. 9.13. Формирование исполнительного адреса при относительной адресации и индексации



Приведенная схема показывает формирование исполнительного адреса (при  $B \neq 0$  и  $X \neq 0$ )

$$A_n = (B) + (X) + D,$$

которое производится в АЛУ процессора или в специальном сумматоре индексной арифметики. В последнем случае повышается быстродействие, но увеличивается объем оборудования ЭВМ.

При индексации код команды, хранящийся в ОП, остается неизменным. Следовательно, одна и та же команда может использоваться при соответствующих изменениях индекса для формирования различных исполнительных адресов.

Индексация служит не только для организации вычислительных циклов, но в сочетании с базированием (иногда такое сочетание называют двухуровневой индексацией) является средством описания элементов информационных массивов, размещаемых в ОП.

Для управления индексацией используются команды, задающие операции над содержимым индексных регистров, — *команды индексной арифметики*. Можно отметить основные виды индексных операций: а) *засылка* в соответствующий индексный регистр начального значения индекса; б) *изменение индекса* и в) *проверка окончания циклических вычислений*.

Изменение индекса состоит в сложении (или вычитании) значения индекса с фиксированным приращением, производимом каждый раз после повторения цикла. Команда изменения индекса указывает номер индексного регистра, а также значение и знак (или адрес) приращения. Если, как это часто бывает, операнды располагаются в последовательных ячейках памяти,



то приращение равно 1, а при побайтной адресации — числу байт  $L$  в слове. В этом случае может использоваться команда изменения с подразумеваемым операндом, т. е. приращение в команде не указывается.

Для проверки того, что требуемое число повторений цикла произведено и следует выйти из цикла, используются или обычная команда условного перехода по условию, налагаемому в этом случае на некоторую переменную, изменяющуюся в процессе вычислений, или специальные команды *Условный переход по счетчику* и *Условный переход по индексу*.

Счетчиком обычно служит один из индексных регистров (общих регистров), в который перед началом вычислительного цикла загружается число повторений цикла. Команда *Условный переход по счетчику* может иметь, например, такой вид

УПСч	$R_1$	$X_2$	$B_2$	$D_2$
------	-------	-------	-------	-------

Эта команда уменьшает на 1 содержимое счетчика (в данном случае регистра с номером  $R_1$ ), и если оно после этого не равно 0, то управление передается по указанному в команде адресу  $(B_2) + D_2 + (X_2)$ . В противном случае управление переходит к следующей по порядку команде (выход из цикла).

Команда *Условный переход по индексу* (если индекс меньше или равен) имеет, например, вид

УПИ	$R_1$	$R_3$	$B_2$	$D_2$
-----	-------	-------	-------	-------

Здесь  $R_1$  — номер индексного регистра;  $R_3$  — номер регистра, хранящего приращение;  $(B_2) + D_2$  — адрес начала цикла. В регистре с номером  $R_3 + 1$  хранится предельное значение индекса. Команда производит суммирование  $(R_1)$  с  $(R_3)$ . Если новое значение индекса меньше или равно содержимому регистра  $R_3 + 1$ , это новое значение помещается в регистр с номером  $R_1$  и происходит переход к команде с адресом  $(B_2) + D_2$  (повторение цикла). В противном случае выполняется следующая за командой УПИ команда (выход из цикла). Аналогично выполняется проверка окончания цикла командой *Условный переход, если индекс больше*.

## 9.9. Теги и дескрипторы. Самоопределяемые данные

Одним из эффективных средств совершенствования архитектуры современных ЭВМ является *теговая организация памяти* [34], при которой каждое хранящееся в памяти (или регистре) слово снабжается указателем — *тегом*, определяющим тип данных (целое двоичное число, число с плавающей точкой, десятичное число, адрес, строка символов, дескриптор и т. д.) (рис. 9.14). В поле тега обычно указываются не только тип, но и длина (формат) и некоторые другие его параметры. Теги формируются компилятором.

Наличие тегов придает хранящимся в машине данным *свойство самоопределяемости*, вносящее принципиальные особенности в архитектуру и функционирование ЭВМ.

Отметим, что ЭВМ с теговой памятью (самоопределяемыми данными) выходит за рамки модели вычислительной машины фон Неймана, исходящей из того, что тип (характер) данного, хранящегося в памяти, определяется только в контексте выполнения программы, точнее говоря, командой, использующей данное в качестве операнда. В обычных ЭВМ, соответствующих классической модели фон Неймана, тип данных-операндов и их формат задаются кодом операции команды, а в ряде случаев размер (формат) операндов определяется специальными полями команды.

Например в ЕС ЭВМ команда *сложение десятичное* своим кодом операции определяет, что адресуемые ею операнды являются десятичными числами. Специальные четырехразрядные поля в этой команде задают число десятичных цифр в операндах. Следует обратить внимание на то, что таким образом в ЕС ЭВМ фактически имеем 256 различных кодов только одной команды *сложение десятичное*.

Теговая организация памяти позволяет достигнуть инвариантности команд относительно типов и форматов операндов, что приводит к значительному сокращению набора команд машины. Это упрощает и делает более регулярной структуру процессора, облегчает работу программиста, в том числе при отладке программ, упрощает компиляторы и сокращает затраты времени на компиляцию (отпадает необходимость выбора типа команды в зависимости от типа данных), облегчает обнаружение ошибок, связанных с некорректным заданием типа данных (например, при попытке сложить адрес с числом с плавающей точкой).

Рис. 9.14. Теговая организация памяти

Тег	Данные
-----	--------

Теговая организация памяти способствует реализации принципа независимости программ от данных.

И наконец, нечто неожиданное. Использование тегов приводит к экономии памяти, так как в программах обычных машин имеется большая информационная избыточность на задание типов и размеров операндов при их использовании несколькими командами.

В качестве недостатка теговой организации памяти можно указать на некоторое замедление работы процессора из-за того,

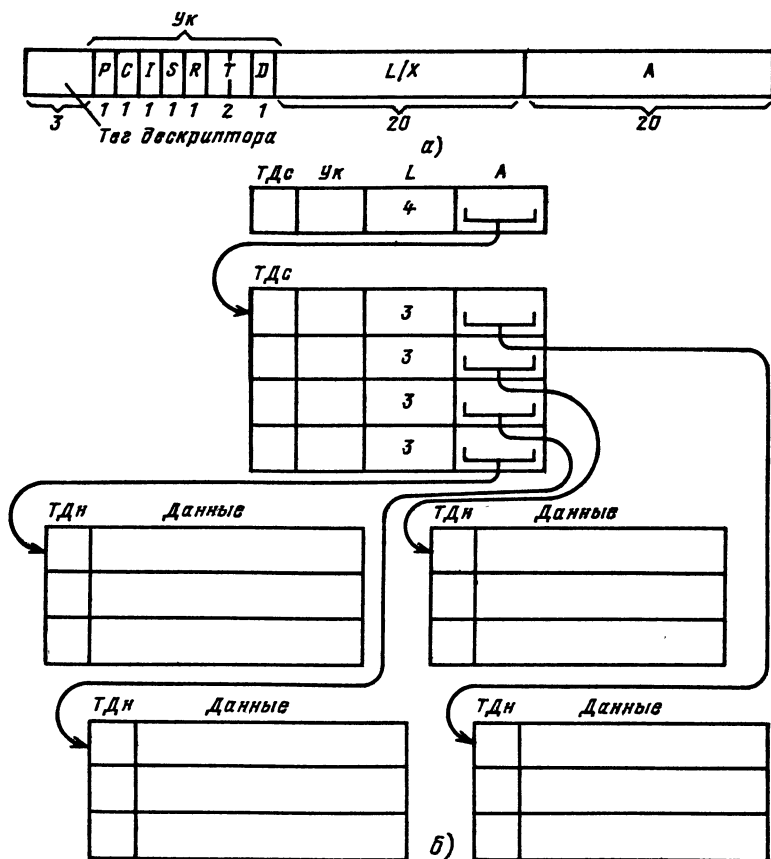


Рис. 9.15. Дескриптор данных ЭВМ B6700 Burroughs (а) и описание двумерного массива данных древовидной структурой дескрипторов (б): ТДс и ТДн — теги дескрипторов и данных

что установление соответствия типа команды типу данных, в обычных ЭВМ выполняемое на этапе компиляции, при использовании тегов переносится на этап выполнения программы.

В архитектуре некоторых ЭВМ используются *дескрипторы* — служебные слова, содержащие описания массивов данных и команд, причем дескрипторы могут употребляться как в машинах с теговой организацией памяти (например, ЭВМ «Эльбрус»), так и без тегов (например, микропроцессор *i APX 432* фирмы Intel). В последнем случае достигается ограниченная самоопределяемость данных.

Дескриптор содержит сведения о размере массива данных, его местоположении (в ОП или внешней памяти), адресе начала массива, типе данных, режиме защиты данных (например, запрет записи в ячейки массива) и некоторых других параметрах данных. Отметим, что задание в дескрипторе размера массива позволяет контролировать выход за границу массива при индексации его элементов. На рис. 9.15, *а* в качестве примера представлен один из видов дескрипторов — дескриптор данных в ЭВМ B6700 фирмы *Borroughs* [34].

Дескриптор содержит специфический тег *ТДс*, указывающий, что данное слово является дескриптором определенного вида, группу указателей *У<sub>к</sub>* и два поля: *А* и *Л/Х*. Поле *А* указывает адрес начала массива данных. В зависимости от значения указателя *І* дескриптор описывает массив данных (*І*=0), и в соответствующее поле помещается длина массива *Л*, или описывает элемент массива (*І*=1), и тогда в поле находится индекс *Х*, указывающий смещение элемента относительно начала массива. «Указатель присутствия» *Р* определяет, находится массив в основной или внешней памяти. В последнем случае поле *А* указывает местоположение массива во внешней памяти. Остальные указатели имеют следующий смысл: *Д* — одинарная или двойная точность представления данных; *Т* — описывается слово или строка; при *Р*=1 данные можно только читать; *С* — указывает на непрерывное или фрагментированное расположение массива в памяти; *С* — является ли дескриптор копией другого дескриптора.

Использование в архитектуре ЭВМ дескрипторов подразумевает, что обращения к информации в памяти производятся через дескрипторы, которые при этом можно рассматривать как дальнейшее развитие аппарата косвенной адресации.

Адресация информации в памяти может осуществляться с помощью цепочки дескрипторов, при этом реализуется многоступенчатая косвенная адресация. Более того, сложные многомерные массивы данных (таблицы и т. п.) эффективно описываются древовидными структурами дескрипторов (рис. 9.15, *б*).

### 9.10. Сопоставление программистских моделей (регистровых структур) машин общего назначения, малых и микроЭВМ и микропроцессоров

Совокупность регистров, доступных программисту и используемых им при составлении программ на языке АССЕМБЛЕР или другом машинно-ориентированном языке, будем называть «программистской моделью ЭВМ». На рис. 9.16 представлены три характерных варианта регистровых структур процессоров. Эти варианты охватывают широкий диапазон — от крупных машин общего назначения до микропроцессоров.

Обращает на себя внимание ставший стандартным в современной архитектуре ЭВМ прием организации сверхоперативной регистровой памяти в виде блока адресуемых (короткими адресами) *общих регистров*, допускающих многоцелевое использование, — для хранения операндов, результата операции, в качестве базовых, индексных регистров и указателей стеков. В машинах с коротким словом, вынуждающим прибегать к одноадресным командам, один из общих регистров выделяется в качестве *аккумулятора* — регистра, в котором находится один из операндов и в который помещается результат операции. Регистр-аккумулятор в явном виде в команде не адресуется — используется подразумеваемая адресация. В машинах, предназначенных для решения сложных вычислительных задач, регистровая адресуемая память включает в себя специальные регистры для операндов и результатов операций с плавающей точкой (*регистры чисел с плавающей точкой*)<sup>1</sup>.

В регистровой структуре процессоров ЕС ЭВМ выделяются 16 общих 32-битных регистра и четыре 64-битных регистра с плавающей точкой. При работе с расширенным 128-битным форматом чисел регистры с плавающей точкой 0,2 и 4,6 попарно объединяются.

Архитектура ЭВМ с коротким словом (малые и микроЭВМ) сформировались позже архитектуры машин общего назначения с достаточно длинным (обычно 32-битным) словом. Отмеченные в предыдущих параграфах трудности, обусловленные коротким словом, стимулировали появление новых архитектурных решений — широкого использования стекового механизма, новых методов адресации. Это нашло отражение в том, что в отличие от машин общего назначения (ЕС ЭВМ) для регистровых структур

---

<sup>1</sup> В малых ЭВМ блок регистров чисел с плавающей точкой часто поставляется по дополнительному заказу. В микропроцессорах эти регистры могут входить в состав схемы отдельного кристалла (корпуса) — арифметического расширителя.

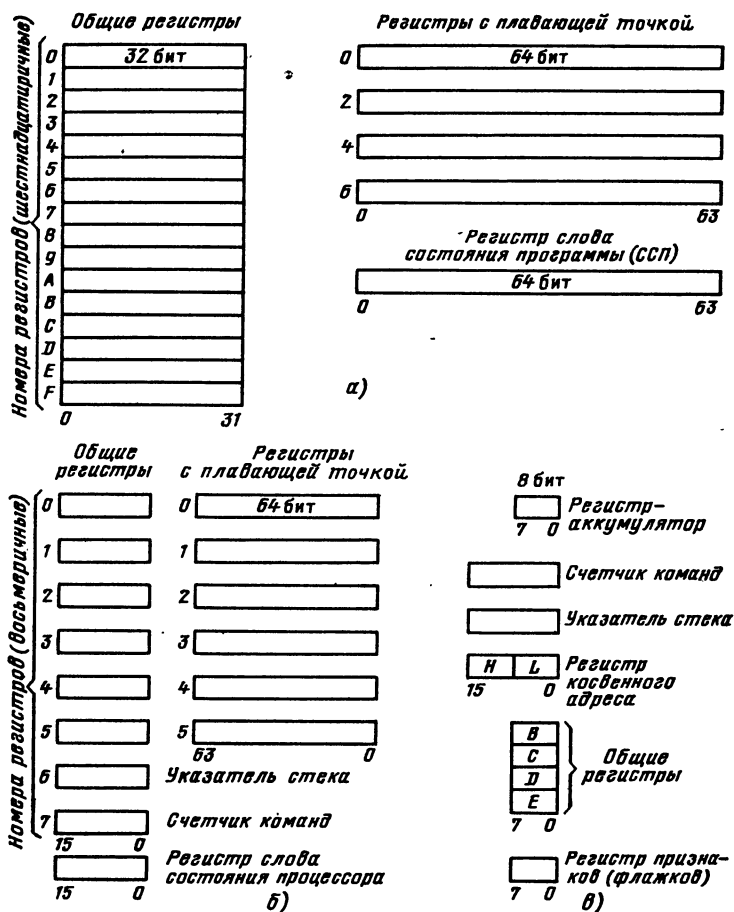


Рис. 9.16. Сопоставление программистских моделей (регистровых структур) машин различного типа:

а — машины общего назначения (ЕС ЭВМ); б — малые и микроЭВМ (СМ-1300, «Электроника-60»); в — микроЭВМ и микропроцессоры (СМ-1800, К580)

малых и микроЭВМ и микропроцессоров характерно наличие регистров — указателей стека (аппаратурного средства реализации стековой памяти)<sup>1</sup> и программно-доступного (адресуемого) регистра счетчика команд, позволяющего реализовывать новые виды адресаций. Для полноты картины следует пред-

<sup>1</sup> Стек в ЕС ЭВМ может организовываться программным путем.

ставленные на рис. 9.16 регистровые структуры рассматривать одновременно с соответствующими адресными структурами ОП на рис. 9.2.

### **9.11. Особенности адресации и системы команд 16-разрядных малых и микроЭВМ**

Архитектура 16-разрядных малых ЭВМ CM-1420 и микроЭВМ CM-1300 и «Электроника-60», подобная архитектуре известной американской серии малых ЭВМ PDP-11, оценивается специалистами как высокоэффективная благодаря высокоразвитой системе адресации, использованию стекового механизма, многорегистровой структуре процессора, переменной длине команд и интерфейсу «общая шина». Она оказала определенное влияние на архитектуру микропроцессоров.

Форматы данных, адресная структура памяти и регистровая структура этих машин представлены на рис. 2.1, 9.2 и 9.16. Машина выполняет операции с 16-битными словами и байтами, которые могут быть двоичными числами со знаком (в дополнительном коде) и без знака, логическими полями, а также с 32- и 64-битными числами с плавающей точкой (при наличии дополнительной аппаратуры).

Существенной особенностью архитектуры является развитая система адресации, придающая системе команд большую гибкость и эффективность. Применяются восемь основных типов адресации, а также стековая адресация при работе с подпрограммами и при прерываниях. Основные типы адресации представлены в табл. 9.2. Возможная в рассматриваемых машинах адресация в командах счетчика команд как общего регистра порождает дополнительные типы адресации (табл. 9.3).

Основная длина команды — одно 16-битное слово. Однако при индексной и непосредственной адресациях, как это следует из табл. 9.2 и 9.3, сразу за командой в следующей ячейке (или следующих двух ячейках) памяти помещается соответственно адресное приращение или непосредственно адресуемый операнд. Поэтому можно считать, что команды имеют три длины: одно, два и три 16-битных слова (16, 32, 48 бит).

При кодировании команд употребляется расширение кода операции. Код операции может иметь длину 4, 7, 8, 10, 12, 13, 16 бит.

На рис. 9.17 представлены основные форматы команд. В отличие от ЕС ЭВМ, где результат операции замещает всегда операнд в первом адресе, в рассматриваемых ЭВМ результат

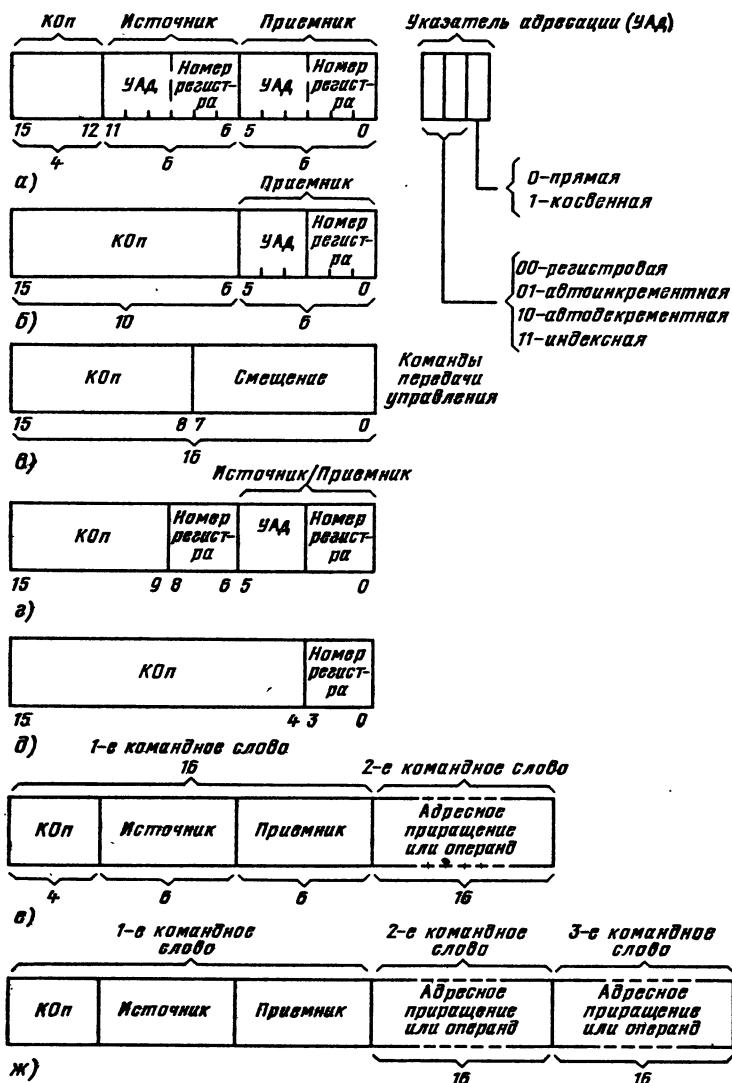


Рис. 9.17. Форматы команд 16-разрядных малых и микроЭВМ

помещается по второму адресу. Поэтому первое адресное поле называется «Адрес источника», или *Источник*, а второе — «Адрес приемника», или *Приемник*.

Адресные поля разбиваются на две части: в правых трех разрядах указывается номер регистра, а в левых — тип адреса-



**Т а б л и ц а 9.2. Основные типы адресации (малая ЭВМ СМ-1420 и микроЭВМ «Электроника-60»)**

УА	Мнемоника	Тип адресации	Описание
0	$Rn$	Регистровая прямая	В $Rn$ (регистре с номером $n$ ) операнд
1	$@ Rn$	Регистровая косвенная	В $Rn$ адрес операнда
2	$(Rn) +$	Автоинкрементная прямая	Содержимое $Rn$ используется как адрес операнда, затем перед выполнением команды увеличивается на 1 (при операциях с байтами) или на 2 (при операциях со словами)
3	$@ (Rn) +$	Автоинкрементная косвенная	Содержимое $Rn$ используется как адрес адреса операнда, затем перед выполнением команды содержимое $Rn$ увеличивается на 2
4	$-(Rn)$	Автодекрементная прямая	Содержимое $Rn$ уменьшается на 1 или 2, затем новое содержимое $Rn$ используется как адрес операнда
5	$@ -(Rn)$	Автодекрементная косвенная	Содержимое $Rn$ уменьшается на 2, затем новое содержимое $Rn$ используется как адрес адреса операнда
6	$X (Rn)$	Индексная прямая	Содержимое $Rn$ (индекс) суммируется с приращением $X$ , находящимся в следующей за первым или вторым словом команды ячейке памяти; полученная сумма используется как адрес операнда. Содержимое $CчК$ (регистра 7) увеличивается на 2 сразу после извлечения $X$
7	$@ X (Rn)$	Индексная косвенная	Содержимое $Rn$ (индекс) суммируется с приращением $X$ , находящимся в следующей за первым или вторым словом команды ячейке памяти; полученная сумма используется как адрес адреса операнда. Содержимое $CчК$ (регистр 7) увеличивается на 2 сразу после извлечения $X$

Примечания: 1. Используется мнемоника ассемблера ЭВМ СМ-1420;  $Rn$  — номер регистра в адресном поле.

2. При индексации значение  $X$  помещается ассемблером во второе или третье слово команды. Если индексную адресацию имеет только один операнд, применяется двухсловная команда со вторым адресным словом. Если оба операнда имеют индексную адресацию, используется трехсловная команда, причем для адресации источника используется второе, а для приемника третье (адресное) слово.

**Т а б л и ц а 9.3. Дополнительные типы адресации при использовании СчК (регистра 7) в качестве общего регистра**

УА	Мнемоника	Используемый основной тип адресации	Получаемый дополнительный тип адресации	Описание
2	$\neq A$	Автоинкрементная прямая с R7	Непосредственная	Операнд A, присоединяемый в виде второго или третьего слова команды, находится в следующей за первым (или вторым) словом команды ячейке памяти, адресуемой СчК. После выборки A содержимое СчК увеличивается на 2
3	@ $\neq A$	Автоинкрементная косвенная с R7	Абсолютная	Содержимое A адресуемой СчК ячейки, следующей за первым (или вторым) словом команды, является адресом адреса операнда. После выборки A содержимое СчК увеличивается на 2
6	A	Индексная прямая с R7	Относительная	Адресом операнда является сумма A содержимого СчК и содержимого ячейки, следующей за первым (или вторым) словом команды
7	@ A	Индексная косвенная с R7	Относительная косвенная	Адресом адреса операнда является сумма A содержимого СчК и содержимого ячейки, следующей за первым (или вторым) словом команды

ции (указатель адресации УА), как это показано на рис. 9.17, а, б, г. Типы адресаций *Источника* и *Приемника* задаются независимо и могут отличаться друг от друга. Команды в зависимости от местоположений операндов, определяющих число обращений к основной памяти, можно разбить на три типа: а) регистр — регистр; б) регистр — память, в) память — память. Следует подчеркнуть, что в команде адрес операнда в памяти не указывается, так как в этом случае используется косвенная адресация. Результат выполнения команд типа регистр — память в зависимости от кода команды может образовываться в регистре или ячейке памяти (в ЕС ЭВМ в подобном случае результат всегда помещается в регистр).

Многие операции могут выполняться как со словами, так и с байтами в зависимости от значения 15-го разряда кода ко-

манды. Двухадресные команды сложения и вычитания выполняют операцию только со словами.

Среди одноадресных отметим команды «Инкремент» и «Декремент», соответственно увеличивающую и уменьшающую содержимое приемника на 1. Имеется большой набор команд условного перехода по результату предыдущей операции: переходы по равенству 0, неравенству 0, плюсу, минусу, переполнению, отсутствию переполнения, переносу, отсутствию переноса, больше или равно 0, меньше 0, больше 0, меньше и равно 0. В командах условного и безусловного переходов (рис. 9.18, *в*) старший байт содержит код операции, а младший — 8-разрядное смещение (число со знаком). Адрес перехода формируется следующим образом: знак смещения копируется в разрядах 8—15, полученное 16-разрядное слово умножается на два и складывается с содержимым счетчика команд, при этом оказывается возможным переход относительно *СчК* до  $177_8$  слов вперед и до  $200_8$  слов назад. Кроме упомянутой команды безусловного перехода имеется команда абсолютного безусловного перехода (имеет формат и способ задания адресации, как у одноадресных команд), позволяющая передавать управление команде в любой ячейке памяти.

Следует обратить внимание на команды и процедуры с использованием механизма стековой памяти. Команда перехода к подпрограмме, имеющая формат, показанный на рис. 8.15, *г*, в поле приемника задает адрес начала подпрограммы, а в разрядах 6—8 — номер регистра *R*, который будет использоваться для хранения адреса возврата к основной программе. Процедура выполнения команды начинается с уменьшения содержания указателя стека на 2 и передачи для сохранения в стек содержимого регистра *R*. Затем в регистр *R* передается содержимое *СчК* (адрес возврата), а в *СчК* поступает адрес приемника (адрес подпрограммы). В команде возврата из подпрограммы (рис. 9.17, *д*) указывается номер регистра, использованного при переходе к подпрограмме для адреса возврата.

При выполнении команды из этого регистра адрес возврата передается в *СчК*, и исходное содержимое регистра восстанавливается из стека.

На рис. 9.17, *е* и *ж* представлены форматы команд, имеющие длину соответственно два и три слова (табл. 9.2 и 9.3).

Сопоставление описанной в настоящем параграфе системы команд с командами ЕС ЭВМ позволяет заключить, что в рассматриваемых малых и микроЭВМ реализуются команды типов RR, RX, RS, SS, SI, что оказывается возможным благодаря наличию в командах полей *УА*, определяющих функции, выполняемые указанными в команде общими регистрами.

## 9.12. Структура команд ЕС ЭВМ

Рассмотрим на примере ЕС ЭВМ структуры команд современных ЭВМ общего назначения. Системы команд этих машин содержат около 200 различных команд, обеспечивающих программирование разнообразных научных, экономических и технических задач высокой степени сложности. Эти системы команд позволяют работать с различными видами и форматами информации (двоичные и десятичные числа, алфавитно-цифровые данные, числа с фиксированной и плавающей точками, слова фиксированной и переменной длины). Используемые в ЕС ЭВМ II очереди форматы информации приведены в гл. 2.

Команды имеют длину в одно, два и три полуслова (2, 4, 6 байт). Результат операции, как правило, помещается на место первого операнда.

В ЕС ЭВМ II очереди имеется шесть основных форматов команд, представленных на рис. 9.18.

1. *Формат «регистр — регистр» RR.* В этом формате представляются короткие двухбайтные команды, выполняющие операции над содержимым общих регистров или регистров с плавающей точкой, номера которых указаны в полях  $R_1$  (первый операнд) и  $R_2$  (второй операнд) команды.

2. *Формат «регистр — индексируемая ячейка» RX.* Команда занимает 32-разрядное слово (4 байта). Первый операнд — содержимое общего регистра (или регистра с плавающей точкой, указанного в поле  $R_1$ ), второй — содержимое ячейки ОП. Адрес этого операнда определяется содержимым базового  $B_2$  и индексного  $X_2$  регистров и смещением  $D_2$ .

3. *Формат «регистр — память» RS.* Команда также занимает слово (4 байта). Поля  $R_1$  и  $R_3$  определяют номера регистров, содержащих первый и третий операнды;  $B_2$ ,  $D_2$  — адрес второго операнда в памяти. Формат RS, в частности, используется для команд групповой передачи данных из группы регистров в группу ячеек ОП и обратно. В этом случае поля  $R_1$  и  $R_3$  задают первый и последний регистры в группе регистров с последовательными номерами, участвующих в передаче.

4. *Формат «память — непосредственный операнд» SI.* Команда занимает слово (4 байта). Адрес первого операнда, находящегося в ОП, задают поля  $B_1$ ,  $D_1$ , а второй операнд  $I_2$  располагается непосредственно в самой команде. Оба операнда имеют длину 1 байт.

5. *Формат S.* Команда занимает слово (4 байта). Первый операнд или его адрес подразумевается, второй находится в ОП.

6. *Формат «память — память» SS.* Команды имеют длину три полуслова (6 байт). Они применяются для операций над

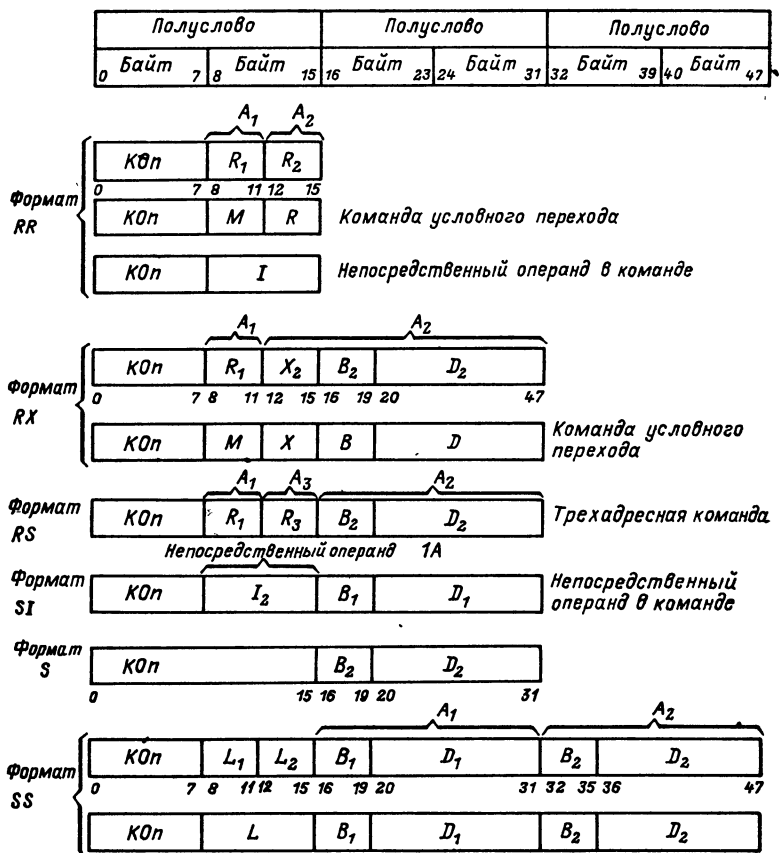


Рис. 9.18. Форматы команд ЕС ЭВМ II очереди

словами переменной длины. Оба операнда располагаются в ОП. Адрес первого операнда (т. е. адрес его левого байта) определяют поля  $B_1$ ,  $D_1$ , а адрес второго операнда —  $B_2$ ,  $D_2$ .

В командах для операций с десятичными числами (первая команда формата SS) в полях  $L_1$  и  $L_2$  указываются длины операндов (16-ричными цифрами). Десятичные операнды могут быть неодинаковой длины (от 1 до 16 байт).

В командах логической обработки данных, представленных словами переменной длины (вторая команда формата SS), оба операнда имеют одинаковую длину (от 1 до 256 байт), задаваемую в поле  $L$  двумя 16-ричными цифрами. Следует заметить, что в командах формата SS длины операндов на единицу превы-

шают значения, указанные в полях  $L_1$ ,  $L_2$ ,  $L$  команды, так как эти поля указывают, сколько байт добавляется к байту, адресуемому исполнительным адресом операнда.

Адрес операнда в ОП образуется суммированием по модулю  $2^{24}$  положительных чисел:

$A_n = (B) + (X) + D$  при наличии индексации;

$A_n = (B) + D$  при отсутствии индексации,

где  $(B)$  — базовый адрес (содержимое 24 младших разрядов общего регистра, номер которого указан в поле  $B$  команды);  $(X)$  — индекс (содержимое 24 младших разрядов общего регистра, указанного в поле  $X$ );  $D$  — смещение.

При нуле в поле  $B$  или  $X$  в качестве соответствующего компонента адреса берется 0, а не содержимое общего регистра с номером 0.

Код операции во всех форматах команд, кроме формата  $S$ , занимает 1 байт (в формате  $S$  2 байта) и задается двумя 16-ричными цифрами, причем первая цифра определяет группу команд для определенного типа и формата данных или управления, а вторая задает команду в этой группе. Нулевой и первый разряды в двоичном представлении  $KOn$  задают формат команды (00 —  $RR$ , 01 —  $RX$ , 10 —  $RS/SI/S$ , 11 —  $SS$ ).

### 9.13. Микропрограммная интерпретация языка команд ЭВМ

В настоящем параграфе в целях подробного описания функционирования процессора воспользуемся *микропрограммным уровнем рассмотрения рабочего процесса ЭВМ* и применим в качестве языка описания язык *микроопераций* (см. гл. 6). Такой подход даст возможность представить *микропрограммирование как способ интерпретации языка команд ЭВМ*.

Чтобы максимально упростить изложение материала, остановимся на предельно простой структуре 8-разрядного процессора (рис. 9.19), напоминающего ранние микропроцессоры. Процессор содержит 8-разрядные *АЛУ*, блок регистров (аккумулятор *Акк*, регистр адреса команды *РзАК*\*, регистр временного хранения *РзВрХр*), основную память *ОП* (ширина выборки 8 бит) с адресным и информационным регистрами *РзАОП* и *РзИОП*, управляющую память (память микропрограмм) *УП* с регистром адреса микрокоманды *РзАМк* и регистром микроко-

---

\* *РзАК* не является счетчиком.

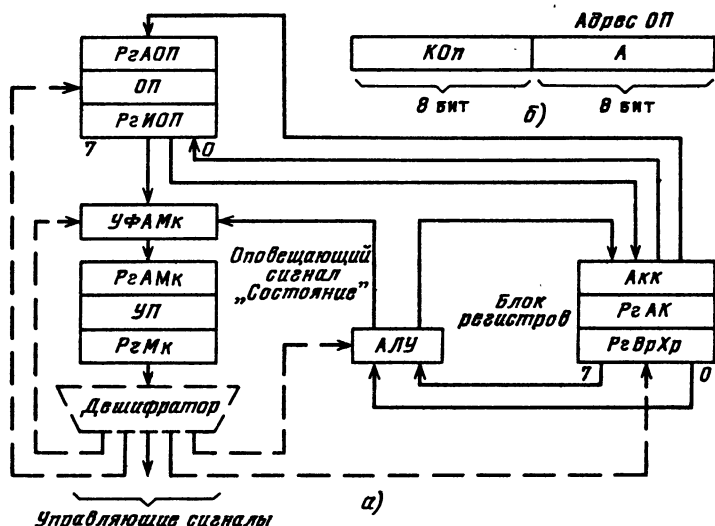


Рис. 9.19. Структура учебного процессора (а) и формат его команды (б)

манд **РгМк**, узел формирования адреса микрокоманды **УФАМк**. На схеме показаны связи между упомянутыми функциональными блоками. Таким образом, введен в рассмотрение некоторый учебный процессор, или микропроцессор [84].

Примем для данного процессора двухбайтный одноадресный формат команды, показанный на рис. 9.19, б. Команда размещается в двух последовательных ячейках ОП.

Реализуем в процессоре следующий набор машинных команд [**А** — адрес 8-битной ячейки ОП, (**А**) — содержимое ячейки]:

Наименование команд	Описание команды
Сложение	$Акк := Акк + (А)$
Вычитание	$Акк := Акк - (А)$
Загрузка	$Акк := (А)$
Запоминание	$(А) := Акк$
Безусловный переход	Передача управления по адресу <b>А</b>
Условный переход	Переход по адресу <b>А</b> , если $Акк = 0$ . Сохранение естественного порядка выборки команд, если $Акк \neq 0$

Приведенные команды реализуются в принятой структуре процессора с помощью, например, следующего набора микроопераций.

1.  $РгИОП := РгZ$ ;
  2.  $РгАОП := РгZ$ ;
  3.  $РгZ := РгИОП$ ;
- Микрооперации межрегистровых передач (**РгZ** — любой регистр из блока регистров)

- |  |   |   |
|--|---|---|
| 4. $P_2Z = Акк;$   | } | Микрооперации управления коммутатором выбора регистра в блоке регистров |
| 5. $P_2Z = P_2AK;$   |   |   |
| 6. $P_2Z = P_2BpXp;$   |   |   |
| 7. $P_2ИОП := ОП [P_2АОП];$  | } | Микрооперации чтения и записи в ОП                                      |
| 8. $ОП [P_2АОП] := P_2ИОП;$  |   |   |
| 9. $Акк := Акк + P_2Z;$  | } | Микрооперации в АЛУ ( $P_2Z$ — любой регистр из блока регистров)        |
| 10. $Акк := Акк - P_2Z;$   |   |   |
| 11. $Акк := P_2Z;$   |   |   |
| 12. $P_2Z := Акк;$   |   |   |
| 13. $Акк := P_2Z + 1;$   |   |   |
| 14. $P_2AMк := P_2AMк + 1;$  | } | Микрооперации формирования адреса следующей микрокоманды                |
| 15. $P_2AMк := P_2ИОП$ $\langle$ в $P_2ИОП$ первый байт команды — код операции $\rangle$ ;                         |   |   |
| или<br>$P_2AMк :=$ результат дешифрации $КОп$ в $P_2ИОП$ ;   |   |   |
| 16. $P_2AMк :=$ Константа $\langle$ в $P_2AMк$ передается содержимое из следующей по порядку ячейки УП $\rangle$ ; |   |   |
| 17. <b>если</b> $Акк=0$ , <b>то</b> $P_2AMк := P_2AMк + 2$<br><b>иначе</b> $P_2AMк := P_2AMк + 1;$                 |   |   |

В микрокомандах могут совмещаться некоторые микрооперации, например в микрокоманде передачи байта из аккумулятора в  $P_2ИОП$  совмещаются микрооперация, иницилирующая эту передачу, и микрооперация выбора аккумулятора из блока регистров.

Перейдем к составлению микропрограмм отдельных процедур и операций. Выполнение любой команды начинается с процедуры выборки команды из ОП по адресу, находящемуся в  $P_2AK$ . Представим эту общую для всех команд процедуру в виде отдельной микропрограммы.

#### *Микропрограмма процедуры выборки команды*

Адрес УП

- 000  $P_2АОП := P_2AK$   $\langle$  загрузка адреса команды в  $P_2АОП$   $\rangle$ ;
- 001  $P_2ИОП := ОП [P_2АОП]$   $\langle$  чтение  $КОп$  (первого байта) команды в  $P_2ИОП$   $\rangle$ ;
- 002  $P_2BpXp := Акк$   $\langle$  сохранение содержимого  $Акк$   $\rangle$ ;
- 003  $Акк := P_2AK + 1$   $\langle$  формирование адреса второго байта команды  $\rangle$ ;
- 004  $P_2AK := Акк$   $\langle$  передача адреса второго байта команды в  $P_2AK$   $\rangle$ ;
- 005  $Акк := P_2BpXp$   $\langle$  восстановление содержимого  $Акк$   $\rangle$ ;
- 006  $P_2AMк :=$  результат дешифрации  $КОп$  в  $P_2ИОП$   $\langle$  формирование в  $P_2AMк$  начального адреса в УП микропрограммы операции, соответствующей  $КОп$  команды  $\rangle$ ;



0 0 0 0 0 0	Микрокоманды регистровых передач
0 0 0 0 0 1	$P_2 ИОП := P_2 Z$
0 0 0 0 1 0	$P_2 Z := P_2 ИОП$
0 0 0 0 1 0	$P_2 АОП := P_2 Z$
0 0 1 0 0 0	Микрокоманды операций в ОП
0 0 1 0 0 1	Чтение
0 0 1 0 1 0	Запись
0 1 0 0 0 0	Микрокоманды АЛУ
0 1 0 0 0 1	$Акк := Акк + P_2 Z$
0 1 0 0 1 0	$Акк := Акк - P_2 Z$
0 1 0 0 1 1	$Акк := P_2 Z$
0 1 0 1 0 0	$P_2 Z := Акк$
0 1 0 1 0 0	$Акк := P_2 Z + 1$
0 1 1 0 0 0	Микрокоманды управления
0 1 1 0 0 1	последовательностью выборки
0 1 1 0 1 0	микрокоманд
0 1 1 0 1 0	$P_2 Амк := дешифрация Коп в P_2 ИОП$
0 1 1 0 1 0	$P_2 Амк := константа$
0 1 1 0 1 0	Ветвления в микропрограмме
0 1 1 0 1 0	по условию $Акк = 0$

Рис. 9.20. Коды микрокоманд учебного процессора

Микропрограмма команды «Сложение»

$P_2 АОП := P_2 АК$  (загрузка адреса операнда);

$P_2 ИОП := ОП [P_2 АОП]$  (чтение адреса операнда);

$P_2 ВрХр := Акк$  (сохранение содержимого Акк);

$Акк := P_2 АК + 1$  (формирование продвинутого адреса команды);

$P_2 АК := Акк$ ;

$Акк := P_2 ВрХр$  (восстановление содержимого Акк);

$P_2 ВрХр := P_2 ИОП$  (загрузка адреса операнда для временного хранения);

$P_2 АОП := P_2 ВрХр$  (передача адреса операнда);

$P_2 ИОП := ОП [P_2 АОП]$  (загрузка операнда);

$P_2 ВрХр := P_2 ИОП$  (чтение операнда из ОП);

$Акк := Акк + P_2 ВрХр$ ; (в Акк образуется сумма);

$P_2 Амк := 0$  (занесение константы 0 — переход к микропрограмме выборки следующей команды);

Микропрограмма команды условного перехода, если  $Акк=0$   
 $P_2АОП := P_2АК$  (загрузка адреса перехода — второго  
 байта команды условного перехода);  
 $P_2ИОП := ОП [P_2АОП]$  (чтение адреса перехода);  
 $P_2ВрХр := Акк$ ;  
 $Акк := P_2АК + 1$ ;  
 $P_2АК := Акк$  (образование в  $P_2АК$  адреса следующей ко-  
 манды при естественном порядке выборки);  
 $Акк := P_2ВрХр$ ;  
 если  $Акк=0$  то  $P_2АМк := P_2АМк + 2$  иначе  $P_2АМк :=$   
 $= P_2АМк + 1$ ;  
 $P_2АМк := 0$  (возврат при  $Акк \neq 0$  к микропрограмме вы-  
 борки команды — сохранение естественного порядка выборки  
 команд);  
 $P_2АК := P_2ИОП$  (загрузка адреса перехода в  $P_2АК$ );  
 $P_2АМк := 0$  (возврат к микропрограмме выборки ко-  
 манды);

Микропрограммы остальных команд читатель может соста-  
 вить сам.

Закончим рассмотрение учебного процессора проектировани-  
 ем формата и кодов микрокоманд. На рис. 9.20 представлен  
 один из возможных вариантов формата и кодов микрокоманд.

## 9.14. Особенности RISC-архитектуры

Развитие архитектуры ЭВМ, направленное на повышение их  
 производительности, во многих случаях идет по пути усложне-  
 ния процессоров путем расширения системы (набора) команд,  
 введения сложных команд, выполняющих процедуры, приближа-  
 ющиеся к примитивам языков высокого уровня, увеличения  
 числа используемых способов адресации и т. д.

Однако расширение и усложнение набора команд порожда-  
 ют и ряд нежелательных побочных эффектов.

Расширение набора команд, увеличение числа способов ад-  
 ресации, введение сложных команд сопровождаются увеличени-  
 ем длины кода команды, в первую очередь, кода операции, что  
 может приводить к использованию «расширяющегося кода опе-  
 рации», увеличению числа форматов команд. Это вызывает ус-  
 ложнение и замедление процесса дешифрации кода операции  
 и других процедур обработки команд. Возрастающая сложность  
 процедур обработки команд заставляет прибегать к микропрог-  
 раммным управляющим устройствам с управляющей памятью  
 вместо более быстродействующих УУ с «жесткой» («схемной»)  
 логикой.

Усложнение процессора делает более трудным или даже невыполнимым реализацию его на одном кристалле интегральной микросхемы, что благодаря сокращению длин межсоединений могло бы облегчить достижение высокой производительности.

Сказанное объясняет, почему в последнее время сформировалось альтернативное по отношению к усложнению архитектуры процессоров направление, использующее при создании сравнительно дешевых высокопроизводительных ЭВМ архитектуру с сокращенным набором команд (СНК-архитектуру), получившую в зарубежной литературе название RISC-архитектуры<sup>1</sup> [58].

*СНК-архитектура* предполагает реализацию в ЭВМ сокращенного набора простейших, но часто употребляемых команд, что позволяет упростить аппаратные средства процессора и благодаря этому получить возможность повысить его быстродействие.

При использовании СНК-архитектуры выбор набора команд и структуры процессора (микропроцессора) направлены на то, чтобы команды набора выполнялись за один машинный цикл процессора. Выполнение более сложных, но редко встречающихся операций обеспечивают подпрограммы.

В ЭВМ с СНК *машинным циклом называют время, в течение которого производится выборка двух операндов из регистров, выполнение операции в АЛУ и запоминание результата в регистре*. Большинство команд в СНК являются быстрыми командами типа «регистр — регистр» и выполняются без обращений к ОП, которые сохраняются лишь в командах загрузки регистров из памяти и запоминания в ОП. Чтобы это было возможным, процессор должен содержать достаточно большое число общих регистров.

Благодаря характерным для СНК-архитектуры особенностям — сокращенному набору команд (обычно не более 50—100), небольшому числу (обычно 2—3) простых способов адресации (в основном регистровой), небольшому числу простых форматов команд с фиксированными размерами и функциональным назначением их полей — упрощается управляющее устройство процессора, который в этом случае обходится без микропрограммного уровня управления и управляющей памяти, и его УУ может быть выполнено на «схемной логике».

Уменьшение количества выполняемых команд и другие отмеченные выше особенности СНК-архитектуры приводят к столь

---

<sup>1</sup> RISC — Reduced Instruction Set Computers — ЭВМ с сокращенным набором команд.

значительному упрощению структуры процессора, что становится возможной его реализация на одном кристалле с оставлением при этом места на кристалле для увеличения до нескольких десятков, а в некоторых машинах даже до сотен числа общих и специализированных регистров.

Большое число регистров, особенно при наличии обеспечивающего их эффективное использование «оптимизирующего компилятора», позволяет до предела сократить обращение к ОП путем сохранения на регистрах промежуточных результатов, передачи через регистры операндов из одних программ в другие программы или подпрограммы, отказа от передач на сохранение в ОП содержимого регистров при прерываниях.

Особенностью СНК-архитектуры является механизм *перекрывающихся регистровых окон*, предназначенный для уменьшения числа обращений к ОП и межрегистровых передач, что способствует повышению производительности ЭВМ.

Процедурам динамически выделяются небольшие группы регистров фиксированной длины (регистровые окна). Окна последовательно выполняемых процедур перекрываются, благодаря чему возможна передача параметров от одной процедуры к другой. При вызове процедуры процессор переключается на работу с другим регистровым окном, при этом не возникает необходимости в передаче содержимого регистров в память.

Окно состоит из трех подгрупп регистров (рис. 9.21). Первая подгруппа содержит параметры, переданные данной процедуре от ее вызвавшей, и результаты для вызывающей процедуры при возврате в нее. Вторая подгруппа содержит локальные переменные процедуры. Третья, являясь буфером для двустороннего обмена между данной и ею вызываемой следующей процедурами, передает последней параметры от данной, которая, в свою

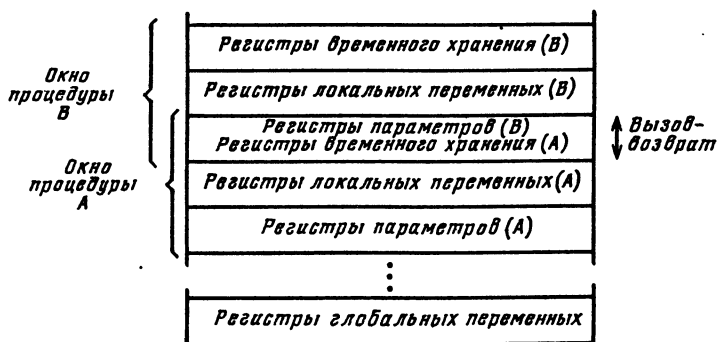


Рис. 9.21. СНК-архитектура ЭВМ. Перекрывающиеся регистровые окна

очередь, получает через этот буфер результаты от ею вызванной процедуры. Таким образом, одна и та же подгруппа для данной процедуры является регистрами временного хранения, а для следующей — регистрами параметров. Отдельное окно, доступное всем процедурам программы, выделяется для ее глобальных переменных.

В настоящее время за рубежом выпущен ряд машин с СНК-архитектурой (RISC-архитектурой). Примером является высокопроизводительный персональный компьютер IBM PC-RT, имеющий 118 команд, всего 2 способа адресации и 2 формата команд, 16 общих регистров, среднее число циклов на команду 3.

Несмотря на начавшееся использование СНК-архитектуры в выпускаемой промышленностью ЭВМ, продолжают дискуссии вокруг достоинств и недостатков этой архитектуры. К последним, в частности, относят большую длину кода программы после компиляции (объектного кода) по сравнению с длиной кода машин обычной архитектуры. Например, при эмуляции команд ЭВМ типа VAX в среднем на каждую его команду требуется пять-шесть команд машины с СНК-архитектурой. Однако, как показали исследования, выигрыш в скорости выполнения команд перекрывает проигрыш от удлинения объектного кода программы.

По последним сведениям (Electronics, 1989, № 3) фирме Intel удалось на основе RISC-архитектуры создать однокристальный микропроцессор 80860, который практически представляет собой кремниевый эквивалент суперЭВМ Gray-1, рассматриваемый в § 15.8.

### **9.15. Понятие о состоянии процессора (программы). Вектор (слово) состояния**

При выполнении процессором программы после каждого рабочего такта, а тем более в результате завершения выполнения очередной команды, изменяется содержимое регистров, счетчиков, состояния отдельных управляющих триггеров. Можно говорить, что изменяется состояние процессора, или, употребляя другую терминологию, *состояние программы*.

Понятие *состояния процессора (состояния программы)* занимает важное место в организации вычислительного процесса в ЭВМ.

Информация о состоянии процессора (программы) лежит в основе многих процедур управления вычислительным процессом, например при анализе ситуаций при отказах и сбоях, при возобновлении выполнения программы после перерывов, вызван-

ных отказами, сбоями, прерываниями, для фиксации состояния процессора (программы) в момент перехода в мультипрограммном режиме от обработки данной программы к другой и т. п.

Состоянием процессора (программы) после данного такта или после выполнения данной команды, строго говоря, следует считать совокупность состояний в соответствующий момент времени всех запоминающих элементов устройства — триггеров, регистров, ячеек памяти.

Однако не вся эта информация исчезает или искажается при переходе к очередной команде или другой программе. Поэтому из всего многообразия информации о состоянии процессора (программы) отбираются наиболее существенные ее элементы, как правило, подверженные изменениям при переходе к другой команде или программе.

Совокупность значений этих информационных элементов получила название *вектора состояния* или *слова состояния процессора (программы)*.

*Вектор состояния* в каждый момент времени должен содержать информацию, достаточную для продолжения выполнения программы или повторного пуска программы с точки, соответствующей моменту формирования данного вектора состояния. При этом предполагается, что остальная информация, характеризующая состояние процессора, например содержимое регистров, или сохраняется, или может быть восстановлена программным путем по копии, сохраненной в памяти.

Вектор состояния формируется в соответствующем регистре

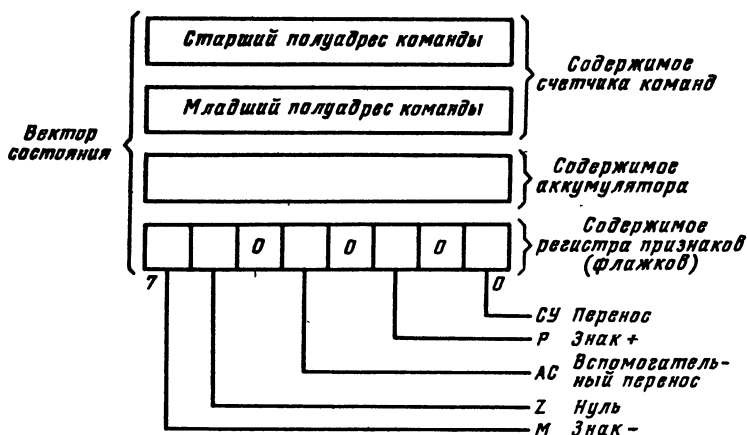


Рис. 9.22. Вектор состояния 8-разрядного микропроцессора К580 (четыре 8-разрядных слова)

(регистрах) процессора, претерпевая изменения после выполнения каждой команды.

Наборы информационных элементов, образующих векторы состояния, отличаются у ЭВМ разных типов. Наиболее просто он выглядит у микропроцессоров. Например, вектор состояния микропроцессора К580, как это показано на рис. 9.22, включает в себя содержимое 16-разрядного счетчика команд (адрес очередной команды), содержимое 8-разрядного регистра признаков, называемое в документации на этот микропроцессор *словом состояния процессора*, и содержимое 8-разрядного аккумулятора АЛУ.

Более крупные ЭВМ, например ЭВМ ЕС, имеют более сложные структуры вектора состояния, или, иначе говоря, слова состояния программы.

Использование слова (вектора) состояния — распространенный прием построения управления устройствами вычислительной техники. Во многих устройствах ЭВМ для организации их функционирования формируются свои, специфические слова состояния (или *байты состояния*), фиксирующие в виде некоторого кода состояние устройства, например готовность его к выполнению задаваемой операции, успешное или неуспешное завершение операции и т. д.

## 9.16. Принципы организации системы прерывания программ

Во время выполнения ЭВМ текущей программы внутри машины и в связанной с ней внешней среде (например, в технологическом процессе, управляемом ЭВМ) могут возникать события, требующие немедленной реакции на них со стороны машины.

Реакция состоит в том, что машина прерывает обработку текущей программы и переходит к выполнению некоторой другой программы, специально предназначенной для данного события. По завершении этой программы ЭВМ возвращается к выполнению прерванной программы.

Рассматриваемый процесс, называемый *прерыванием программ*, поясняется на рис. 9.23. Принципиально важным является то, что моменты возникновения событий, требующих прерывания программ, заранее неизвестны и поэтому не могут быть учтены при программировании.

Каждое событие, требующее прерывания, сопровождается сигналом, оповещающим ЭВМ. Назовем эти сигналы *запросами прерывания*. Программу, затребованную запросом прерывания, назовем *прерывающей программой*, противопоставляя ее пре-

рываемой программе, выполнявшейся машиной до появления запроса.

Запросы на прерывания могут возникать внутри самой ЭВМ и в ее внешней среде. К первым относятся, например, запросы при возникновении в ЭВМ таких событий, как появление ошибки в работе ее аппаратуры, переполнение разрядной сетки, попытка деления на 0, выход из установленной для данной программы области памяти, затребование периферийным устройством операции ввода-вывода, завершение операции ввода-вывода периферийным устройством или возникновение при этой операции особой ситуации и др. Хотя некоторые из указанных событий порождаются самой программой, моменты их появления, как правило, невозможно предусмотреть. Запросы во внешней среде могут возникать от других ЭВМ, от аварийных и некоторых других датчиков технологического процесса и т. п.

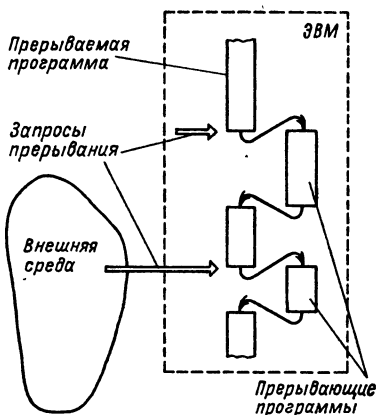


Рис. 9.23. Прерывание программы

В сущности, запросы прерывания генерируются несколькими развивающимися параллельно во времени процессами, которые в некоторые моменты требуют вмешательства процессора. К этим процессам, в частности, относятся процесс выполнения самой программы, процесс контроля правильности работы ЭВМ, операции ввода-вывода, технологический процесс в управляемом машиной объекте и др.

Возможность прерывания программ — важное архитектурное свойство ЭВМ, позволяющее эффективно использовать производительность процессора при наличии нескольких протекающих параллельно во времени процессов, требующих в произвольные моменты времени управления и обслуживания со стороны процессора. В первую очередь это относится к организации параллельной во времени работы процессора и периферийных устройств машины, а также к использованию ЭВМ для управления в реальном времени технологическими процессами.

В некоторых машинах наряду или вместо прерывания с переключением управления на другую программу используется примитивное прерывание — так называемая приостановка, когда по соответствующему запросу приостанавливается выполнение программы и выполняется аппаратурными средствами некоторая



процедура без изменения содержания счетчика команд, а по ее окончании продолжается выполнение приостановленной программы.

Чтобы ЭВМ могла, не требуя больших усилий от программиста, реализовывать с высоким быстродействием прерывания программ, машине необходимо придать соответствующие аппаратные и программные средства, совокупность которых получила название *системы прерывания программ* или *контроллера прерывания*.

Основными функциями системы прерывания являются: запоминание состояния прерываемой программы и осуществление перехода к прерывающей программе;

восстановление состояния прерванной программы и возврат к ней.

При наличии нескольких источников запросов прерывания должен быть установлен определенный порядок (дисциплина) в обслуживании поступающих запросов. Другими словами, между запросами (и соответствующими прерывающими программами) должны быть установлены *приоритетные соотношения*, определяющие, какой из нескольких поступивших запросов подлежит обработке в первую очередь, и устанавливающие, имеет право или не имеет данный запрос (прерывающая программа) прерывать ту или иную программу. Приоритетный выбор запроса для исполнения входит в процедуру перехода к прерывающей программе.

*Характеристики системы прерывания.* Для оценки эффективности систем прерывания могут быть использованы следующие характеристики.

Общее число запросов прерывания (входов в систему прерывания).

Время реакции — время между появлением запроса прерывания и началом выполнения прерывающей программы.

На рис. 9.24 приведена упрощенная временная диаграмма процесса прерывания в предположении, что управление запо-

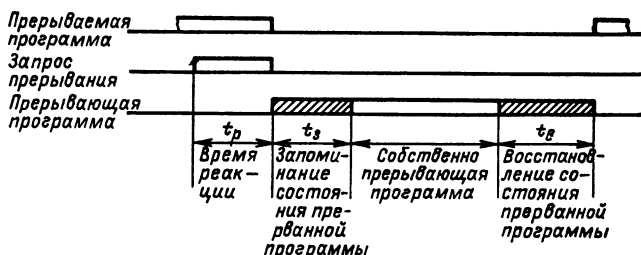


Рис. 9.24. Упрощенная временная диаграмма процесса прерывания

минанием состояния и возвратом возложено на саму прерывающую программу, которая в этом случае состоит из трех частей: подготовительной и заключительной, обеспечивающих переключение программ, и собственно прерывающей программы, выполняющей затребованную запросом работу.

Для одного и того же запроса задержки в исполнении прерывающей программы зависят от того, сколько программ со старшим приоритетом ждут обслуживания. Поэтому время реакции определяют для запроса с наивысшим приоритетом.

Время реакции зависит от того, в какой момент допустимо прерывание. Большей частью прерывание допускается после окончания текущей команды. В этом случае время реакции определяется в основном длительностью выполнения команды.

Это время реакции может оказаться недопустимо большим для ЭВМ, предназначенных для работы в реальном масштабе времени. В таких машинах часто допускается прерывание после любого такта выполнения команды. Однако при этом возрастает количество информации, подлежащей запоминанию и восстановлению при переключении программ, так как в этом случае необходимо сохранять также и состояния в момент прерывания счетчика тактов, регистра кода операции и некоторых других. Поэтому такая организация прерывания возможна только в машинах с быстродействующей сверхоперативной памятью.

Имеются ситуации, в которых желательно немедленное прерывание. Если аппаратура контроля обнаружила ошибку, то целесообразно сразу же прервать операцию, пока ошибка не оказала влияния на следующие такты работы машины.

Затраты времени на переключение программ (издержки прерывания) равны суммарному расходу времени на запоминание и восстановление состояния программы:

$$t_{\text{изд}} = t_3 + t_{\text{в}}.$$

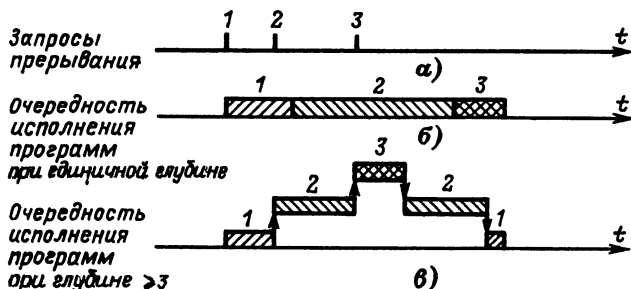


Рис. 9.25. Прерывание в системах с различной глубиной прерывания

Г л у б и н а п р е р ы в а н и я — максимальное число программ, которые могут прерывать друг друга. Если после перехода к прерывающей программе и вплоть до ее окончания прием других запросов запрещается, то говорят, что система имеет глубину прерывания, равную 1. Глубина равна  $n$ , если допускается последовательное прерывание до  $n$  программ. Глубина прерывания обычно совпадает с числом уровней приоритета в системе прерываний. На рис. 9.25, *a* — *в* показано прерывание в системах с различной глубиной прерывания (предполагается, что приоритет каждого следующего запроса выше предыдущего). Системы с большим значением глубины прерывания обеспечивают более быструю реакцию на срочные запросы.

Если запрос окажется необслуженным к моменту прихода нового запроса от того же источника, то возникнет так называемое *насыщение системы прерывания*. В этом случае предыдущий запрос прерывания от данного источника будет машиной утрачен, что недопустимо. Быстродействие ЭВМ, характеристики системы прерывания, число источников прерывания и частота возникновения запросов должны быть согласованы таким образом, чтобы насыщение было невозможным.

Ч и с л о к л а с с о в (у р о в н е й) п р е р ы в а н и я. В ЭВМ число различных запросов (причин) прерывания может достигать нескольких десятков или сотен. В таких случаях часто запросы разделяют на отдельные классы или уровни.

Совокупность запросов, инициирующих одну и ту же прерывающую программу, образует *класс или уровень прерывания* (рис. 9.26).

Запросы всех источников прерывания поступают на регистр запросов прерывания *РзЗП*, устанавливая соответствующие его

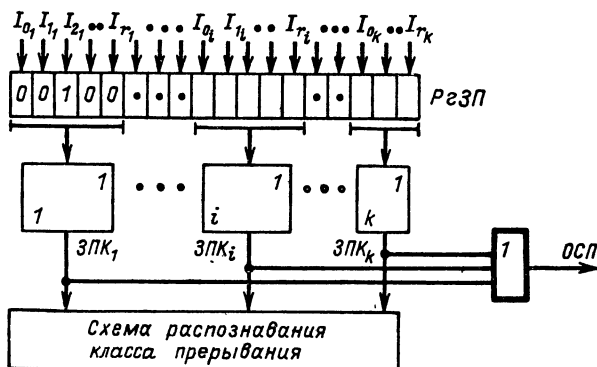


Рис. 9.26. Разделение запросов на классы прерывания

разряды (флажки) в состояние 1, указывающее на наличие запроса прерывания определенного источника. Запросы классов прерывания  $ЗПК_1 — ЗПК_k$  формируются элементами ИЛИ, объединяющими разряды  $РзЗП$ , относящиеся к соответствующим классам (уровням). Еще одна схема ИЛИ формирует общий сигнал прерывания  $ОСП$ , поступающий в устройство управления процессора. Значение сигнала  $ОСП$  определяется выражением

$$ОСП = \bigvee_{i=1}^k ЗПК_i = \bigvee_{i=1}^k \left( \bigvee_{j=0}^{r_i} I_j \right).$$

Информация о действительной причине прерывания, породившей запрос данного класса, содержится в коде прерывания, который отражает состояние разрядов  $РзЗП$ , относящихся к данному классу прерывания. После принятия запроса прерывания на исполнение и передачи управления прерывающей программе соответствующий триггер  $РзЗП$  сбрасывается. Объединение запросов в классы прерывания позволяет уменьшить объем аппаратуры, но связано с замедлением работы системы прерывания.

*Организация перехода к прерывающей программе. Приоритетное обслуживание запросов прерывания.* Назовем вектором прерывания вектор начального состояния прерывающей программы. Вектор прерывания содержит всю необходимую информацию для перехода к прерывающей программе, в том числе ее начальный адрес. Каждому запросу (уровню) прерывания, а в ряде случаев, например в малых и микроЭВМ и микропроцессорах, каждому периферийному устройству соответствует свой вектор прерывания, способный инициировать выполнение соответствующей прерывающей программы. Векторы прерывания обычно находятся в специально выделенных фиксированных ячейках памяти.

Главное место в процедуре перехода к прерывающей программе занимают передача из соответствующего регистра (регистров) процессора в память (в частности, в стек) на сохранение текущего вектора состояния прерываемой программы (чтобы можно было вернуться к ее исполнению) и загрузка в регистр (регистры) процессора вектора прерывания прерывающей программы, к которой при этом переходит управление процессором.

Процедура организации перехода к прерывающей программе включает в себя выделение из выставленных запросов такого, который имеет наибольший приоритет.

Различают *абсолютный и относительный приоритеты*. Запрос, имеющий абсолютный приоритет, прерывает выполняемую

программу и инициирует выполнение соответствующей прерывающей программы. Запрос с относительным приоритетом является первым кандидатом на обслуживание после завершения выполнения текущей программы.

Если наиболее приоритетный из выставленных запросов прерывания не превосходит по уровню приоритета выполняемую процессором программу, то запрос прерывания игнорируется или его обслуживание откладывается до завершения выполнения текущей программы.

Простейший способ установления приоритетных соотношений между запросами (уровнями) прерывания состоит в том, что приоритет определяется порядком присоединения линий сигналов запросов ко входам системы прерывания. При появлении нескольких запросов прерывания первым воспринимается запрос, поступивший на вход с меньшим номером. В этом случае приоритет является жестко фиксированным. Изменить приоритетные соотношения можно лишь пересоединением линий сигналов запросов на входах системы прерывания.

*Процедура прерывания с опросом источников (флажков) прерывания.* При указанном способе задания приоритета между запросами каждому источнику запросов соответствует разряд (флажок) в регистре запросов прерывания (*регистре флажков*).

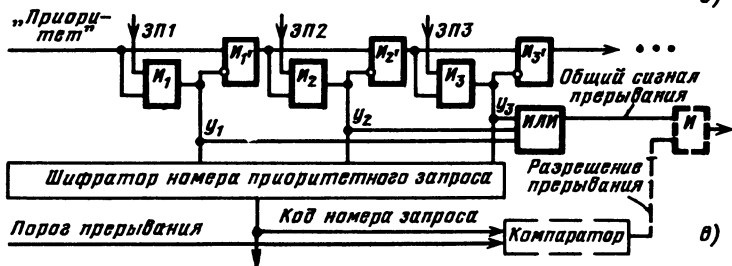
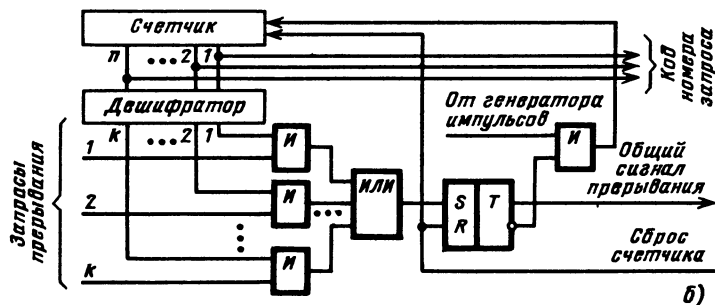
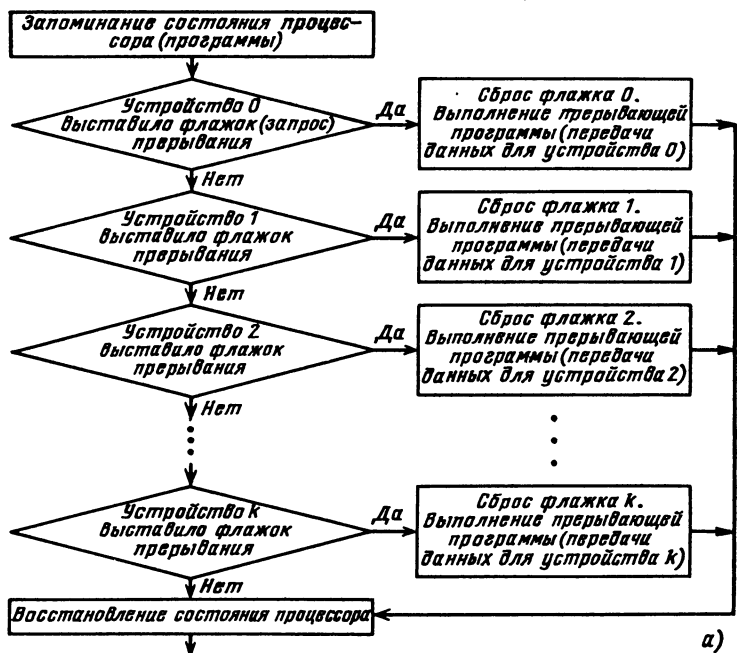
При наличии запроса или нескольких запросов прерывания формируется общий сигнал прерывания (ОСП) (как это, например, показано на рис. 9.26), инициирующий выполняемую специальной программой или аппаратурой процедуру опроса регистра прерывания (флажков) или просто линий сигналов прерывания для установления источника, выставившего запрос прерывания наибольшего приоритета. По существу, эта процедура состоит в определении местоположения крайней слева единицы (крайнего флажка) в регистре запросов прерывания.

На рис. 9.27 приведены различные способы реализации процедуры опроса источников сигналов прерывания. На рис. 9.27, а показан процесс прерывания с программным опросом флажков прерывания (или, другими словами, опросом периферийных устройств, затребовавших передачу данных). Программный опрос источников прерываний занимает сравнительно много времени. Для уменьшения этого времени процедуру опроса реализуют аппаратным путем.

*Схема циклического опроса запросов (источников) прерываний* (рис. 9.27, б). Опрос  $k$  линий запросов прерывания (или

---

Рис. 9.27. Способы опроса источников сигналов прерывания:  
а — программный опрос; б — циклический (многотактный) опрос; в — цепочный одноктактный опрос («дейзи-цепочка»)



разрядов регистра запросов прерывания) производится последовательно (циклически) с помощью  $n$ -разрядного счетчика ( $2^n \geq k$ ), на который с некоторой частотой поступают импульсы от генератора. Поиск приоритетного запроса прерывания начинается со сброса счетчика и одновременно триггера  $T$  в нулевое состояние, при этом импульсы генератора начинают поступать на вход счетчика. При помощи дешифратора и элементов  $И$  в каждом такте поиска проверяется наличие запроса прерывания, номер которого совпадает с кодом счетчика. Если на данном входе нет запроса прерывания, то после прибавления 1 к счетчику проверяется следующий по порядку вход. Если имеется запрос, триггер  $T$  перебрасывается в 1, при этом в процессор посылается общий сигнал прерывания  $ОСП$  и прекращается поступление импульсов на вход счетчика, т. е. завершается цикл просмотра входов системы прерывания. Содержимое счетчика — код номера старшего по приоритету выставленного запроса — используется для формирования начального адреса прерывающей программы. После передачи управления прерывающей программе счетчик (и триггер  $T$ ) сбрасывается в 0, и процедура опроса запросов возобновляется, начиная с первого входа.

Циклический (последовательный) опрос входов системы прерывания в аппаратурном отношении сравнительно прост, однако время реакции и при этом методе все-таки велико, особенно при большом числе источников запросов. Поэтому во многих случаях, например в ряде микропроцессоров, предназначенных для использования при работе в реальном времени, применяют схемы, позволяющие определять номер выставленного запроса или уровня прерывания старшего приоритета за один такт.

*Цепочечная одноктактная схема определения приоритетного запроса («дейзи-цепочка»)* представлена на рис. 9.27, в. Как и в предыдущих случаях, приоритет запросов прерывания возрастает с уменьшением их номера.

Процедура определения приоритетного запроса инициируется сигналом *Приоритет*, поступающим на цепочку последовательно включенных схем  $И$ . При отсутствии запросов этот сигнал пройдет через цепочку и сигнал общего запроса прерывания не сформируется. Если среди выставленных запросов прерывания наибольший приоритет имеет  $i$ -й запрос, то распространение сигнала *Приоритет* правее схемы  $И$  с номером  $i$  блокируется. На  $i$ -м выходе цепочечной схемы будет сигнал  $y_i = 1$ , на всех других 0. В процессор поступит общий сигнал прерывания, при этом дешифратор по сигналу  $y_i = 1$  сформирует код номера  $i$ -го запроса, принятого к обслуживанию. По сигналу процессора *Подтверждение прерывания* (на рис. 9.27 не показан) этот код передает-

ся в процессор и используется для формирования начального адреса прерывающей программы.

Схемы, представленные на рис. 9.27, б и в, производят поиск крайней левой единицы в наборе сигналов прерывания и формируют код номера  $i$  запроса, удовлетворяющего условию

$$3\Pi_i \left( \bigwedge_{j=1}^{i-1} \overline{3\Pi_j} \right) = 1.$$

### *Векторное прерывание*

Представленные на рис. 9.27 способы определения запроса с наибольшим приоритетом включают в себя так или иначе выполняемую процедуру опроса источников прерывания (входов системы прерывания). Эта процедура, даже если она выполняется аппаратными средствами, требует сравнительно больших временных затрат.

Более гибким и динамичным является *векторное прерывание*, при котором исключается опрос источников прерывания (флажков регистра прерывания).

Прерывание называется векторным, если источник прерывания, выставляя запрос прерывания, посылает в процессор (выставляет на шины интерфейса) код адреса в памяти своего вектора прерывания.

Отметим, что если прерывание на основе опроса источников прерываний всегда сопровождается переходом по одному и тому же адресу и инициирует одну и ту же прерывающую подпрограмму, которая после идентификации источника запроса и формирования адреса начала соответствующей запросу прерывающей программы передает ей управление, то при векторном прерывании каждому запросу прерывания, или, другими словами, устройству — источнику прерывания, соответствует переход к начальному адресу соответствующей прерывающей программы, задаваемому вектором прерывания.

### *Программно-управляемый приоритет прерывающих программ*

Относительная степень важности программ, их частота повторения, относительная степень срочности в ходе вычислительного процесса могут меняться, требуя установления новых приоритетных отношений. Поэтому во многих случаях приоритет между прерывающими программами не может быть зафиксирован раз и навсегда. Необходимо иметь возможность изменять по мере надобности приоритетные соотношения программным путем, другими словами, приоритет между прерывающими программами должен быть динамичным, т. е. программно-управляемым.



В ЭВМ широко применяются два способа реализации программно-управляемого приоритета прерывающих программ, в которых используются соответственно *порог прерывания* (в малых и микроЭВМ) и *маски прерывания* (в ЭВМ общего назначения).

**Порог прерывания.** Этот способ позволяет в ходе вычислительного процесса программным путем изменять уровень приоритета процессора (а следовательно, и обрабатываемой в данный момент на процессоре программы) относительно приоритетов запросов источников прерывания (в основном периферийных устройств), другими словами, задавать порог прерывания, т. е. минимальный уровень приоритета запросов, которым разрешается прерывать программу, идущую на процессоре.

Порог прерывания задается командой программы, устанавливающей в регистре порога прерывания *код порога прерывания*. Специальная схема выделяет наиболее приоритетный запрос прерывания, сравнивает его приоритет с порогом прерывания и, если он оказывается выше порога, вырабатывает общий сигнал прерывания, и начинается процедура прерывания (рис. 9.27, в).

В современных ЭВМ общего назначения наибольшее распространение получило программное управление приоритетом на основе маски прерывания (рис. 9.28).

**Маска прерывания** представляет собой двоичный код, разряды которого поставлены в соответствие запросам или классам прерывания. Маска загружается командой программы в регистр маски. Состояние 1 в данном разряде регистра маски разрешает, а состояние 0 запрещает (*маскирует*) прерывание текущей программы от соответствующего запроса. Таким образом, програм-

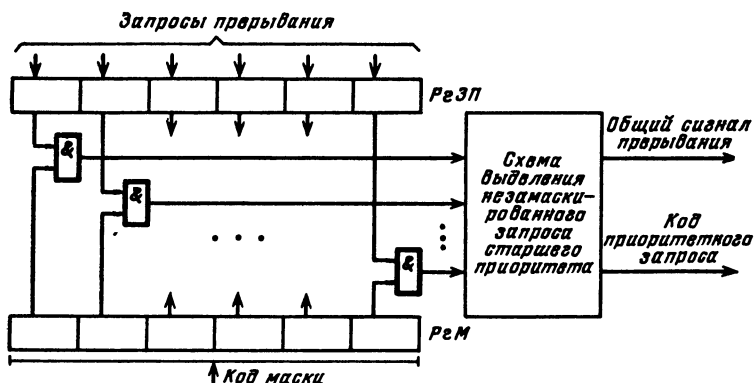


Рис. 9.28. Программно-управляемый приоритет на основе маски прерывания

ма, изменяя маску в регистре маски, может устанавливать произвольные приоритетные соотношения между программами без перекоммутации линий, по которым поступают запросы прерывания. Каждая прерывающая программа может установить свою маску. При формировании маски 1 устанавливаются в разряды, соответствующие запросам (прерывающим программам) с более высоким, чем у данной программы, приоритетом.

Схемы И выделяют поступившие незамаскированные запросы прерывания, из которых специальная схема, аналогичная цепочечной схеме на рис. 9.28, в, выделяет наиболее приоритетный и формирует код его номера  $i$ , удовлетворяющего условию

$$\left( \bigwedge_{j=1}^{i-1} \overline{3P_j P_2 M_j} \right) 3P_i P_2 M_i = 1.$$

С замаскированным запросом в зависимости от причины прерывания поступают двояким образом: или он игнорируется, или запоминается, с тем чтобы осуществить затребованные действия, когда запрет будет снят. Например, если прерывание вызвано окончанием операции в периферийном устройстве, то его следует, как правило, запомнить, так как иначе ЭВМ останется неосведомленной о том, что периферийное устройство освободилось. Прерывание, вызванное переполнением разрядной сетки при арифметической операции, следует при его маскировании игнорировать, так как запоминание этого запроса может привести к тому, что он окажет действие на часть программы или другую программу, к которым это переполнение не относится.

## 9.17. Особенности систем прерывания малых ЭВМ

Во многих малых ЭВМ, микропроцессорах и построенных на микропроцессорах микроЭВМ реализованы многоуровневые векторные системы прерывания с порогом прерывания и с использо-



Рис. 9.29. Вектор состояния процессора в малых ЭВМ (СМ-1420, СМ-1300 и др.)

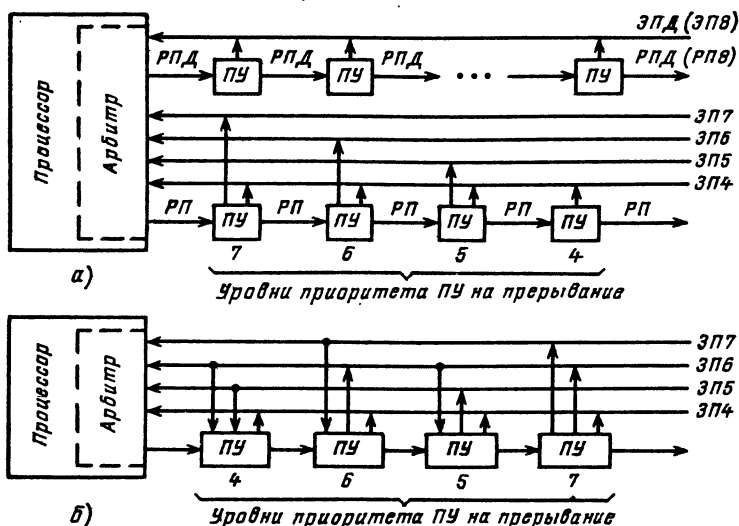


Рис. 9.30. Схема цепей запросов и разрешений прерываний в малых и микроЭВМ с интерфейсом «Q-шина»:

а — с позиционно-зависимым приоритетом; б — с позиционно-независимым приоритетом; ЗП<sub>i</sub>, РП<sub>i</sub> — соответственно запрос и разрешение прерывания *i*-го класса; ЗПД, РПД — соответственно запрос и разрешение прямого доступа к памяти

ванием стековой памяти в процедурах перехода к прерывающей программе и возврата к прерванной программе.

### Векторная система прерывания в малых машинах СМ ЭВМ

Структура вектора состояния, характерная для малых машин (СМ-1420, СМ-1300 и др.), показана на рис. 9.29. Такова же структура вектора прерывания.

Рассмотрим особенности системы прерывания малых и микроЭВМ, в которых используется интерфейс «Q-шина» (см. гл. 11).

**Запросы прерываний.** Запросы внешних прерываний генерируются периферийными устройствами, подсоединенными к интерфейсу «Q-шина». На рис. 9.30 представлены варианты схем присоединения периферийных устройств и процессора (схемы арбитража) к линиям сигналов запросов и разрешения прерывания и прямого доступа к памяти. Имеется четыре уровня приоритета запросов прерывания — с четвертого по седьмой (в порядке возрастания). Еще более высокий (восьмой) уровень приоритета имеют запросы прямого доступа к памяти. Каждый

уровень обслуживает своя линия запросов прерывания  $ЗП_i$ , к которой параллельно (по схеме ИЛИ) подсоединяются ПУ соответствующего уровня приоритета. Имеется одна линия для выдаваемого арбитром сигнала разрешения прерывания РП, проходящая последовательно через все ПУ с приоритетом от четвертого до седьмого<sup>1</sup>. Кроме того, имеется отдельная линия для сигнала разрешения прямого доступа к памяти РПД, также проходящая последовательно через все ПУ, подключенные к линии запросов прямого доступа ЗПД.

При наличии одной линии разрешения прерывания для выделения устройства, которому разрешается прерывание, используется цепочечный метод (см. § 9.16), при этом возможны два варианта схем прерывания: с позиционно-зависимым (рис. 9.30, а) и с позиционно-независимым приоритетом (рис. 9.30, б). В позиционно-зависимой схеме устройства подсоединяются к процессору, точнее, к линии РП в порядке убывания приоритета. Если это неудобно, может применяться позиционно-независимая схема, в которой благодаря дополнительным связям при появлении  $ЗП$  на линиях более высокого приоритета выставившие  $ЗП$  устройства меньшего приоритета игнорируют сигнал разрешения прерывания и пропускают его на соседние устройства<sup>2</sup>. Во второй схеме позиционность сохраняется только в отношении устройств, имеющих одинаковый приоритет. Из них преимущественное право на прерывание имеет устройство, расположенное электрически ближе к процессору.

Схема Арбитра из выставленных запросов выделяет запрос старшего уровня приоритета и сравнивает его уровень с приоритетом процессора, т. е. с программно-устанавливаемым в регистре слова состояния процессора *порогом прерывания* (может принимать значения 4—7). Если уровень наиболее приоритетного из выставленных запросов прерывания превышает порог прерывания, арбитр (процессор) после завершения выполнения текущей команды выдает сигнал разрешения прерывания на линию РП. Этот сигнал поступает в первое по пути его прохождения выставившее запрос (и не заблокированное в схеме рис. 9.30, б) устройство, которое прекращает дальнейшее распространение сигнала РП.

Устройство, пославшее  $ЗП_i$  и получившее разрешение на

---

<sup>1</sup> Для обеспечения совместимости с одноплатными микроЭВМ (например, «Электроника-60»), имеющими упрощенную (одноуровневую) систему прерывания, ПУ уровней приоритета 5—7 выставляют  $ЗП$  также и на линию  $ЗП_4$ .

<sup>2</sup> Выставление устройством уровня приоритета 7 запроса также на линию  $ЗП_6$  упрощает арбитражное решение на уровнях 4 и 5.

прерывание, передает в процессор адрес соответствующего вектора прерывания. Процессор, получив адрес вектора прерывания, помещает в стек, т. е. в ячейки памяти, адресуемые указателем стека, два слова вектора состояния: сначала текущее слово состояния процессора (второе слово вектора состояния), затем первое слово — содержимое счетчика команд (продвинутый адрес прерванной программы). Перед каждой передачей в стек значение указателя стека уменьшается на два.

Далее в счетчик команд из ячейки, хранящей первое слово вектора прерывания, передается начальный адрес прерывающей программы, а из следующей ячейки второе слово вектора прерывания заносится в регистр слова состояния процессора. В новом слове состояния процессора порог прерывания должен быть не меньше уровня приоритета принятого к обслуживанию запроса, чтобы повторный запрос от этого источника прерывания не мог прервать выполняемую прерывающую программу. Управление переходит к программе обработки прерывания, заданной вектором прерывания. Если эта программа использует общие регистры, то она начинается с передачи их содержимого в стековую память с помощью команд передачи с автодекрементной прямой адресации по регистру указателя стека.

Возврат к прерванной программе осуществляет заключительная часть прерывающей программы, в которой команды передачи данных с автоинкрементной прямой адресацией по указателю стека производят передачу из стека сохраненных в нем состояний общих регистров в соответствующие регистры. Последней командой прерывающей программы — командой «Возврат из прерываний» — первое слово вектора состояния прерванной программы загружается из стека в счетчик команд, а второе слово — в регистр слова состояния процессора. Передача каждого слова сопровождается увеличением УС на два. После этого восстанавливается выполнение прерванной программы.

Имеются особенности в процедуре выполнения запросов прерываний *ЗП8* (запросов прямого доступа к памяти). Их приоритет всегда выше приоритета процессора. Поэтому в ответ на запрос *ЗПД (ЗП8)* сигнал разрешения *РПД* посылается немедленно, даже если не завершено выполнение текущей команды<sup>1</sup>, и производится обмен данными между периферийным устройством и *ОП* без участия процессора.

---

<sup>1</sup> Выдача сигнала *РПД* может быть задержана до завершения процессором начатой передачи слова через шину интерфейса, а также до завершения команды операции с плавающей точкой, если для этого потребуется не более установленного максимального времени задержки.

## 9.18. Система прерывания и некоторые особенности организации режимов управления в ЕС ЭВМ

В машинах общего назначения ЕС ЭВМ возможны два режима: *режим основного управления (режим ВС)*, соответствующий режиму работы машин ЕС ЭВМ I очереди и полностью программно-совместимый с ними; *режим расширенного управления (режим ЕС)*, позволяющий использовать новые аппаратные и программные средства и соответственно новые функциональные возможности, появившиеся в ЕС ЭВМ II очереди.

Для реализации режима расширенного управления в состав процессора введены 16 32-разрядных управляющих регистров и используются дополнительные фиксированные ячейки из постоянно распределенной области памяти. В режиме расширенного управления возможно расширение системы прерывания, реализуются регистрация программных событий, развитая система отсчета времени, средства обеспечения мониторинговых программ, режим виртуальной памяти, блоковое мультиплексирование в каналах ввода-вывода и другие дополнительные функции.

В ЕС ЭВМ II очереди имеются следующие классы (уровни) прерывания: 1) от ввода-вывода; 2) программные; 3) при обращении к супервизору; 4) внешние; 5) от схем контроля и 6) повторного пуска (только в режиме ЕС).

В каждом классе имеется несколько источников (причин) запросов прерывания.

*Прерывания от ввода-вывода*, поступающие из каналов, указывают на завершение (может быть и неудачное) операций в канале или периферийном устройстве.

*Программные прерывания* являются результатом ошибок в программе (использование несуществующего кода операции или адреса, нарушение защиты памяти, различные виды переполнений разрядной сетки, исчезновение порядка при арифметических операциях и др.).

*Прерывание при обращении к супервизору* происходит, если в программе встречается команда «Обращение к супервизору». Этот вид прерывания является средством, позволяющим пользователю инициировать работу супервизора для выполнения определенных действий (например, получить для программы пользователя дополнительную область памяти, пустить операцию ввода-вывода и др.).

*Внешние прерывания* вызываются сигналами, поступающими от кнопки прерывания на пульте оператора, от датчика времени (таймера), от внешних по отношению к машине объектов.

*Прерывания от схем контроля* возникают, если обнаруживаются ошибки в работе оборудования машины.

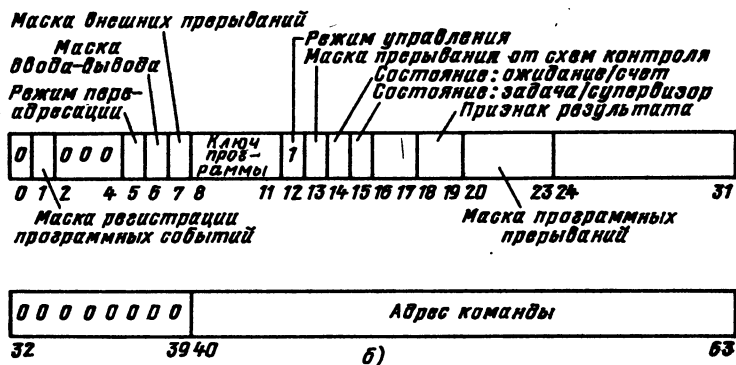
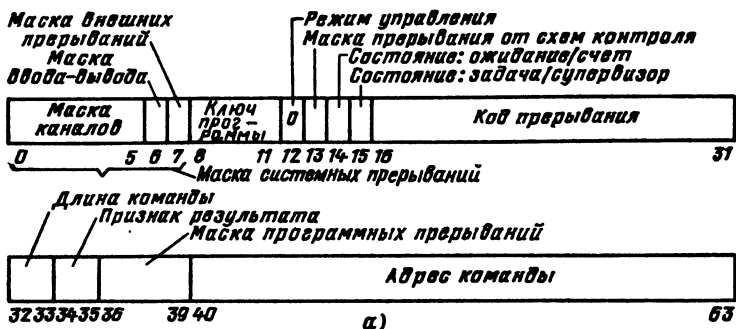


Рис. 9.31. Структура ССП в машинах ЕС ЭВМ:

а — режим основного управления; б — режим расширенного управления

Прерывания повторного пуска поступают от кнопки повторного пуска на пульте оператора.

Система прерывания в ЕС является комбинированной — с аппаратурным выделением наиболее приоритетного из запросов от классов прерывания и программно-опросным выявлением конкретной причины прерывания внутри класса по «коду прерывания», который запоминается в ОП при каждом прерывании, а также по дополнительной информации, запоминаемой в ОП при некоторых видах прерывания (например, по слову состояния канала при прерываниях от ввода-вывода). Особую гибкость этой системе прерывания придают маски прерывания.

Текущее состояние программы характеризуется вектором состояния, который в ЕС ЭВМ называется *словом состояния программы* (ССП).

В ЭВМ ЕС, архитектура которых сложилась в основном раньше, чем получили развитие малые и микроЭВМ, отсутствует

характерный для этих машин механизм стековой памяти, и ячейки памяти с фиксированными адресами используются для хранения векторов состояний прерванных и векторов прерываний прерывающих программ (которые в данном случае называются *старыми и новыми словами состояния программ*).

На рис. 9.31 представлена структура ССП для обоих режимов управления. Слово состояния программы занимает два 32-разрядных слова.

Режим управления устанавливается значением 12-го разряда ССП. При 0 в этом разряде имеет место режим основного управления, при 1 — расширенного.

Рассмотрим значения полей ССП основного режима управления.

Непосредственно к системе прерывания относятся поля ССП, содержащие маски прерывания и код прерывания. Предусмотрены следующие маски:

*маска системных прерываний.* Отдельные разряды этой маски имеют следующий смысл: 0 — маска мультиплексного канала; 1—6 — маска селекторных каналов соответственно № 1—5; 6 — определяет, допустимо ли прерывание для каналов с номерами выше 5 (прерывание возможно, если равны 1 этот разряд и разряд маски соответствующего канала в управляющем регистре); 7 — маска внешних прерываний. Замаскированные прерывания ввода-вывода и внешние сохраняются в ожидании последующей обработки;

*маска прерывания от схем контроля* (разряд 13). Замаскированное прерывание теряется;

*маска программных прерываний.* Отдельные разряды этой маски имеют следующий смысл: 36 — маска переполнения с фиксированной точкой; 37 — маска десятичного переполнения; 38 — маска исчезновения порядка; 39 — маска потери значимости (нулевая мантисса). Замаскированные прерывания теряются.

Маски устанавливаются и изменяются программным путем. Прерывание данного класса разрешается при наличии 1 в соответствующем разряде маски и запрещается (маскируется), если этот разряд содержит 0. Установка масок производится либо в момент прерывания в соответствии с содержанием нового ССП, либо при выполнении специальной команды загрузки ССП.

*Код прерывания* (разряды 16—31) указывает для данного класса прерывания фактическую причину (источник) запроса прерывания. Например, при прерывании от ввода-вывода этот код указывает источник запроса прерывания (номер канала, номер периферийного устройства), при программном прерывании — причину прерывания (некорректность кода операции, ад-



ресаии, представления данных, переполнение разрядной сетки, исчезновение порядка, потеря значимости и др.).

В остальных полях ССП содержится информация, характеризующая другие параметры состояния программы в момент прерывания. Сюда относятся присвоенный программе код ключа защиты памяти и код признака результата последней выполнявшейся команды. Слово состояния программы содержит адрес команды, с которой программа должна начать работать при новом пуске, а также код длины предшествующей команды. Это позволяет в ряде случаев определить ее адрес, что используется при некоторых прерываниях (например, от программных ошибок).

Слово состояния программы хранит также *программные состояния* процессора для данной программы, определяемые состоянием соответствующих управляющих триггеров.

В рассматриваемых ЭВМ существуют четыре альтернативных программных состояния процессора:

1) *Стоп или Работа* (переключение производится вручную). В состоянии *Стоп* команды программы и прерывания не выполняются, электронные часы ЭВМ стоят. В состоянии *Работа* команды и прерывания могут выполняться;

2) *Ожидание или Счет* (определяется значением разряда 14 ССП). В состоянии *Ожидание* программа ожидает прерывания (например, от ввода-вывода), команды не выполняются, машинные часы работают. В состоянии *Счет* команды выполняются обычным путем;

3) *Супервизор или Задача* (определяется значением разряда 15 ССП). В состоянии *Супервизор* могут выполняться все команды, а в состоянии *Задача* не выполняются команды ввода-вывода и некоторые команды управления (привилегированные команды);

4) *Прерывание разрешено или Прерывание замаскировано*. Прерывание может быть замаскировано установкой нулей в разряды масок прерываний. Если какие-либо разряды этих масок содержат единицу, то соответствующие прерывания выполняются (состояние *Прерывания разрешено*).

В режиме расширенного управления введены новые прерывания. К ним относятся программные прерывания, связанные: а) с работой монитора (программа сбора статистических показателей о работе программ и процессора); б) с регистрацией программных событий; в) с динамической переадресацией (неправильное использование сегмента, страницы, нарушения спецификации при переадресации). Введены новые внешние прерывания для режима многопроцессорной системы (оповещение о сбое, экстренный сигнал и сигнал внешнего вызова) и от систе-

мы отсчета времени (от таймера процессора, компаратора и при нарушении синхронизации часов). Потребовалось ввести новые маски и коды прерывания. Поэтому в режиме ЕС расширен состав и изменено расположение полей в ССП — часть полей ССП вынесена в управляющие регистры и ячейки постоянно распределенной области памяти. Поля с кодом прерывания и кодом длины команды располагаются в фиксированных ячейках этой области памяти.

В управляющем регистре 0 разряд 0 задает (запрещает) блок-мультиплексный режим каналов ввода-вывода; разряд 1 — управление запретом установки маски системы; разряд 2 — управление синхронизацией часов; разряды 8, 9 и 11 — управление размером страниц и сегментов при динамическом распределении памяти; разряды 16—19 — маски оповещения о сбое, экстренном сигнале, сигнале внешнего вызова.

В управляющем регистре 1 разряды 0—7 и 8—25 выделены для указания соответственно длины и адреса таблицы сегментов. Расширенные маски каналов находятся в разрядах 0—31 управляющего регистра 2. Маска монитора находится в разрядах 16—31 управляющего регистра 8. Расширенная маска программных событий размещена в разрядах 0—3 и 16—31 управляющего регистра 9. В управляющих регистрах 10 и 11 в разрядах 8—31 находятся граничные адреса области памяти, на которую распространяется регистрация программных событий.

Процедура перехода к прерывающей программе и возврата из нее, реализуемая в ЕС ЭВМ, пояснена на рис. 9.32. Каждому

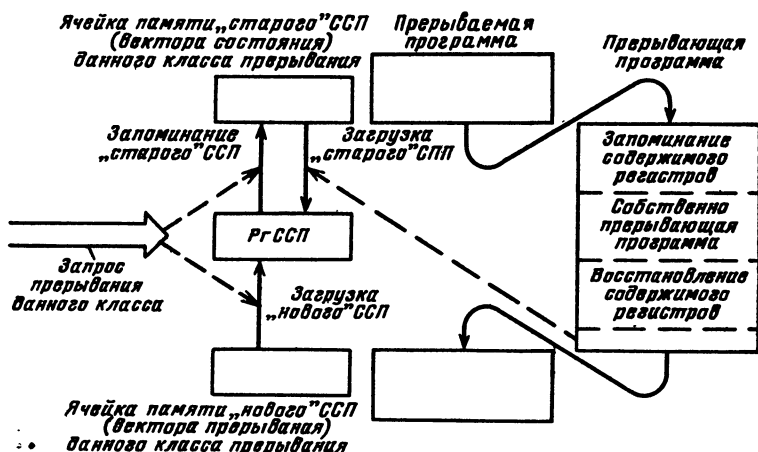


Рис. 9.32. Процедура перехода к прерывающей программе и возврат из нее в ЕС ЭВМ

классу прерывания отводятся в постоянно распределенной области памяти фиксированные ячейки ОП для хранения «старого» ССП и «нового» ССП.

При выполнении текущей программы ее ССП находится в регистре текущего ССП *РгССП* и участвует в управлении вычислительным процессом, при этом отдельные поля ССП (например, счетчик команд) нужным образом изменяются. В ячейках «новых» ССП для всех классов прерывания хранятся ССП, содержащие информацию, достаточную для начала функционирования соответствующих прерывающих программ.

При поступлении запроса от данного класса прерывания, если этот запрос не маскирован в маске прерывания, аппаратурные средства выполняют следующую процедуру. В соответствующую группу разрядов *РгССП* (режим *ВС*) или соответствующую ячейку постоянно распределенной области памяти (режим *ЕС*) записывается код прерывания, содержащий информацию о конкретной причине прерывания. Затем ССП из *РгССП* передается в предусмотренную для данного класса прерывания ячейку ОП для «старого» ССП, а из ячейки «нового» ССП для этого класса «новое» ССП (ССП прерывающей программы, содержащее в том числе ее начальный адрес) загружается в *РгССП*. С этого момента управление переходит к прерывающей программе. На время замены ССП прерывания от любых классов запрещены.

Выполнение прерывающей программы начинается с запоминания в памяти содержимого тех общих регистров и регистров плавающей точки, которые ею будут использованы. Далее выполняется собственно прерывающая программа, которая начинается с анализа «кода прерывания» и определения конкретной причины прерывания и инициирования подпрограммы отработки прерывания, соответствующей этой причине.

Заключительная часть прерывающей программы восстанавливает сохраненное в ОП содержимое регистров и завершает свою работу командой загрузки слова состояния прерванной программы из ячейки «старого» ССП в *РгССП*. Управление переходит к прерванной программе.

Следует различать приоритет между запросами прерывания разных классов и приоритет между прерывающими программами. Первый устанавливает лишь очередность восприятия запросов, поступивших одновременно, а второй, более важный — старшинство (степень срочности) в выполнении прерывающих программ разных классов, другими словами, определяет, имеет ли право данная прерывающая программа прервать выполняемую в данный момент программу.

Установлен следующий порядок приоритета между запро-

сами прерывания: 1) прерывания от схем контроля; 2) прерывания повторного пуска; 3) программные прерывания или прерывания при обращении к супервизору (запросы этих прерываний не могут возникать одновременно); 4) внешние прерывания; 5) прерывания от ввода-вывода.

При обработке прерывания от схемы контроля прием других запросов прерывания не производится. Следовательно, этот класс прерываний имеет наивысший приоритет не только между запросами прерывания, но и между прерывающими программами. Для других классов прерывания приоритет между прерывающими программами имеет порядок, противоположный порядку приоритета между запросами.

Фактический порядок исполнения прерывающих программ определяется приоритетом не между запросами прерывания, а между прерывающими программами [22а].

Построение и функционирование системы прерывания в машинах ЕС ЭВМ поясняются схемой на рис. 9.33 [18]. Работа системы прерывания управляется общими синхросигналами процессора. Запросы прерывания поступают в регистр запросов

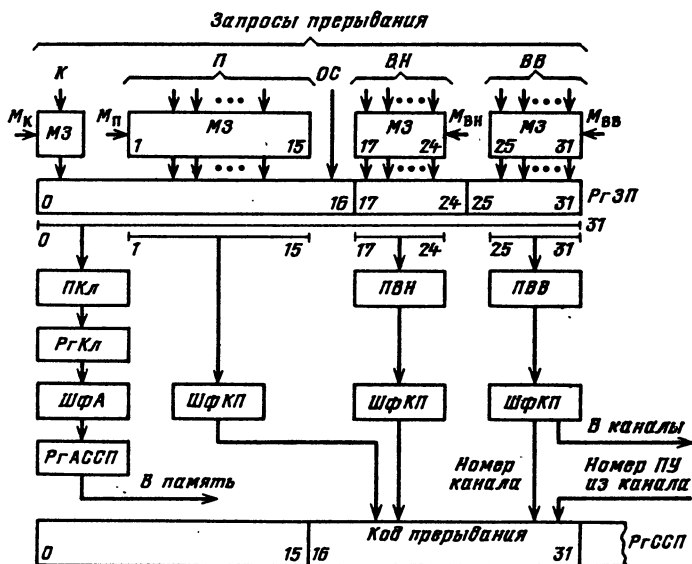


Рис. 9.33. Структура системы прерывания (режим основного управления):

запросы прерывания: **К** — от схем контроля; **П** — программные; **ОС** — обращение к супервизору; **ВН** — внешние; **ВВ** — ввода-вывода; маски: контроля  $M_K = P_2$  ССП [13]; программы  $M_P = P_2$  ССП [36—39]; внешних прерываний  $M_{VN} = P_2$  ССП [7]; ввода-вывода  $M_{VB} = P_2$  ССП [0—6]

прерывания  $P_2ЗП$  через схемы маскирования запросов  $MЗ$ , состоящие из элементов И, управляемых сигналами триггеров соответствующих разрядов масок в  $P_2ССП$ . Схема приоритетного выбора запроса класса прерывания  $ПКл$ , анализируя немаскированные запросы, устанавливает 1 в соответствующий разряд регистра класса прерывания  $P_2Кл$ , в котором разряды 0—5 присвоены отдельным классам в порядке убывания приоритетов запросов прерывания.

Для принятого к обслуживанию класса прерывания в соответствующее поле  $P_2ССП$  передается код прерывания, формируемый шифратором кода прерывания  $ШФКП$  непосредственно из кода, стоящего в соответствующей классу группе разрядов  $P_2ЗП$  при программных прерываниях, и по выходу схемы приоритетного выбора источника запроса прерывания при прерываниях от ввода-вывода и внешних (схемы  $ПВВ$  и  $ПВн$ ). При прерываниях от ввода-вывода в поле кода прерывания  $P_2ССП$  поступает от соответствующего  $ШФКП$  номер канала, а из самого канала — номер периферийного устройства. Заметим, что при прерываниях от схем контроля код прерывания состоит из одних 0.

По коду в  $P_2КЛ$  шифратор адреса  $ШФА$  формирует в  $P_2АССП$  фиксированный адрес ячейки «старого»  $ССП$  данного класса прерывания, куда передается  $ССП$  текущей программы из  $P_2ССП$ . Затем установкой 1 в старший разряд  $P_2АССП$  образуется фиксированный адрес ячейки, из которой «новое»  $ССП$  загружается в  $P_2ССП$ , после чего управление переходит к прерывающей программе.

## 9.19. Процедура выполнения команд. Рабочий цикл процессора

Функционирование процессора в основном состоит из повторяющихся *рабочих циклов*, каждый из которых соответствует выполнению одной команды программы. Завершив рабочий цикл для текущей команды, процессор переходит к выполнению рабочего цикла для следующей команды программы.

На рис. 9.34 представлена схема рабочего цикла процессора [17]. Хотя эта схема отражает некоторые особенности функционирования процессоров ЕС ЭВМ, тем не менее она имеет достаточно общий характер.

На схеме показаны варианты рабочего цикла для четырех групп команд: 1) основных (осуществляющих арифметические, логические и пересылочные операции); 2) передачи управления; 3) ввода-вывода и 4) системных (устанавливающих состояние

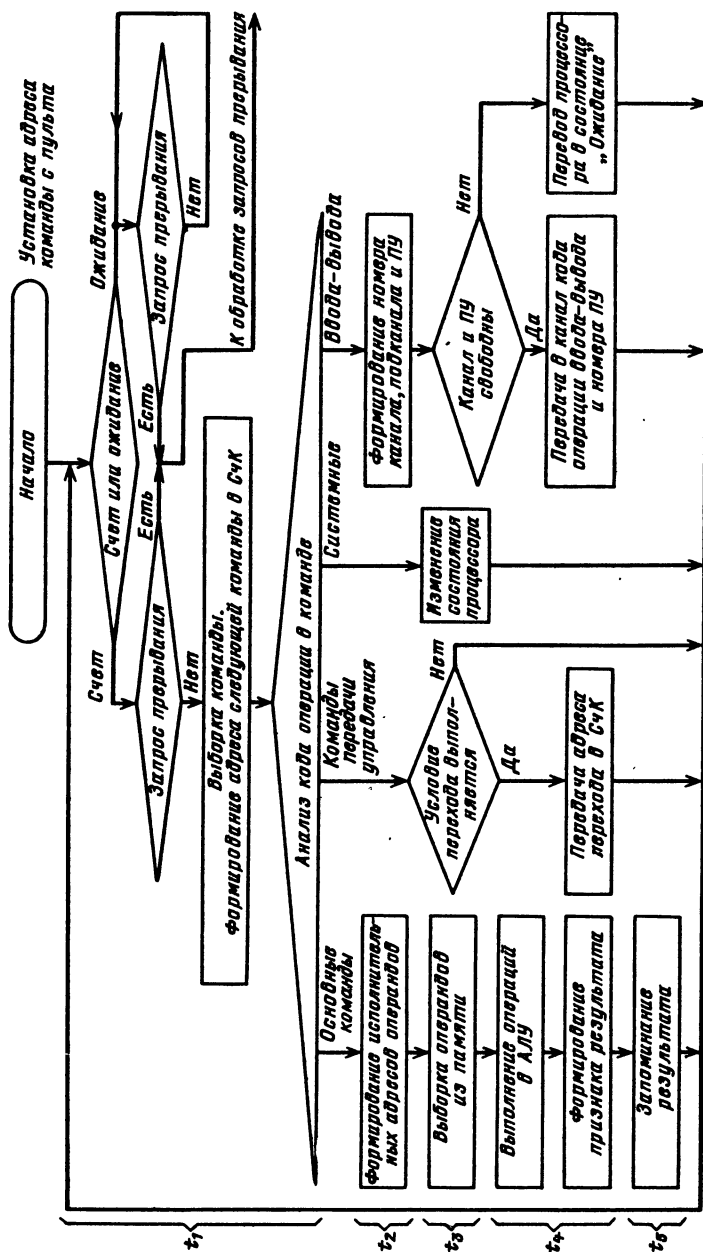


Рис. 9.34. Рабочий цикл процессора

процессора, маску прерывания, слово состояния программы и др.).

Рабочий цикл начинается с распознавания состояния процессора. Устанавливается, какое из альтернативных состояний — *Счет* или *Ожидание* — имеет место. Далее проверяется наличие незамаскированных прерываний.

В состоянии *Ожидание* никакие программы не выполняются. Процессор ждет прихода запроса прерывания, после чего управление переходит к соответствующей прерывающей программе, переводящей процессор в состояние *Счет*.

В состоянии *Счет* при наличии незамаскированных прерываний происходит выход из нормального рабочего цикла и переход к процедуре обработки запросов прерывания.

При отсутствии в состоянии *Счет* запросов прерывания последовательно выполняются этапы рабочего цикла: выборка очередной команды и определение по коду операции команды ее группы, подготовка операндов (формирование исполнительных адресов и выборка операндов из памяти), обработка операндов в АЛУ и запоминание результата.

На этапе выборки очередной команды образуется согласно естественному порядку адрес следующей за ней команды (продвинутый адрес), при этом содержимое счетчика команд (соответствующего поля ССП) увеличивается на число, равное числу байт в очередной команде. В некоторых ЭВМ формирование адреса следующей команды составляет отдельный этап, завершающий рабочий цикл.

В процессе выполнения заданной командой операции формируется *признак результата операции*, используемый командами условного перехода при организации ветвлений в программах.

Указанная выше последовательность этапов составляет основной вариант рабочего цикла, реализуемый при выполнении основных команд.

При выполнении команд передачи управления проверяется заданное командой (например, ее полем маски) условие. Если условие не выполняется, то следующую команду указывает продвинутый адрес, ранее установленный в *СчК* (регистре ССП). Если условие выполняется или имеется один из вариантов команды безусловного перехода, то адрес, задаваемый командой передачи управления, передается в *СчК*.

Команды ввода-вывода инициируют в канале операцию обмена информацией между ядром ЭВМ (основной памятью) и периферийным устройством. Сама эта операция выполняется каналом под управлением его собственной программы (см. гл. 11). Поэтому на долю процессора остается только процедура опроса состояний канала и периферийного устройст-

ва —, свободны ли они для операции ввода-вывода. Если свободны, процессор выдает в канал информацию, необходимую для начала операции ввода-вывода. В противном случае процессор переключается в состоянии *Ожидание* и ждет сигнала прерывания от этого канала.

Системные команды осуществляют переключения состояния процессора (программы) путем загрузки нового ССП или его части. В частности, эти команды изменяют маски прерывания, устанавливают ключи памяти и ключи защиты в ССП, реализуют операции прямого управления.

## 9.20. Принцип совмещения операций академика С. А. Лебедева. Конвейер операций

Вернемся к схеме рабочего цикла (рис. 9.34) и рассмотрим совокупность этапов цикла для основных команд (основной вариант цикла). Если эти этапы выполняются последовательно во времени, то, суммируя обозначенные на рисунке продолжительности отдельных этапов, получаем время цикла

$$t_{\text{посл}} = t_1 + t_2 + t_3 + t_4 + t_5$$

и производительность процессора, операций (команд)/с,

$$P_{\text{посл}} = 1/t_{\text{посл}} = 1/(t_1 + t_2 + t_3 + t_4 + t_5).$$

Во многих случаях последовательная процедура выполнения этапов цикла не обеспечивает требуемую производительность процессора.

Академик С. А. Лебедев в 1956 г. предложил повышать производительность, используя *принцип совмещения во времени отдельных операций (этапов) рабочего цикла*, и реализовал этот принцип в ЭВМ М-20 в форме параллельного выполнения во времени операции в АЛУ и выборки из памяти следующей команды.

Пусть рабочий цикл процессора состоит из  $k$  этапов, причем  $i$ -й этап имеет продолжительность  $t_i$ , тогда при последовательном выполнении этапов продолжительность процедуры

$$t_{\text{посл}} = \sum_{i=1}^k t_i \quad (9.5)$$

и общая производительность процессора, операций/с,



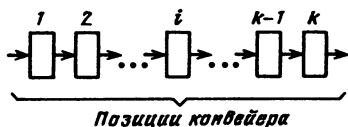


Рис. 9.35. Конвейер операций

$$P_{\text{посл}} = 1 / \sum_{i=1}^k t_i \quad (9.6)$$

Скорость работы машины может быть увеличена, если для выполнения каждого этапа иметь отдельный аппаратный блок и соединить эти блоки в обрабатывающую линию — конвейер операций (в данном случае конвейер команд) так, чтобы результат выполнения в данном блоке некоторого этапа передавался для реализации очередного этапа на следующий блок, и т. д. (рис. 9.35).

*Синхронный конвейер операций* [29]. Если конвейер работает в принудительном темпе и для выполнения любого этапа выделено одно и то же время  $t_r$  (такт конвейера), то такой конвейер называется синхронным.

Разбиение процедуры на этапы и выбор длительности такта производится согласно условиям

$$t_r = \max \{t_i\}, \quad i = 1, \dots, k; \quad (9.7)$$

$$t_i + t_{i+1} > t_r, \quad i = 1, \dots, k, \quad (9.8)$$

причем в силу цикличности рабочего процесса в последнем неравенстве принимаем  $t_{k+1} = t_1$ .

Если для каких-либо смежных этапов второе условие не выполняется, то их следует объединить в один этап либо наиболее длинный этап разбить на несколько этапов. В последнем случае заново выбирается  $t_r$  и вновь проверяется условие (9.7).

На рис. 9.36 показана временная диаграмма выполнения команд на 5-позиционном синхронном конвейере. Одинаковыми символами помечены разные этапы рабочего цикла одной и той же команды.

После того как все позиции конвейера окажутся заполненными, параллельно во времени обрабатывается столько команд, сколько в конвейере обрабатывающих блоков (позиций).

Конвейер характеризуется коэффициентом совмещения операций, равным числу одновременно выполняемых этапов обработки информации.

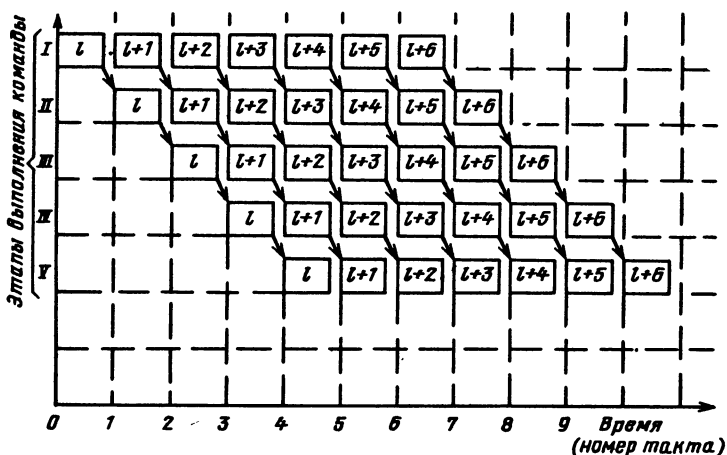


Рис. 9.36. Синхронный конвейер команд

Номинальная производительность синхронного конвейера при его полной загрузке

$$P_{\text{конв}}^{\text{ном}} = 1/t_{\tau}. \quad (9.9)$$

Найдем соотношение производительностей процессора при конвейерной обработке и при последовательном выполнении этапов рабочего цикла.

Из (9.5) и (9.7) имеем

$$kt_{\tau} \geq \sum_{i=1}^k t_i = t_{\text{посл}}, \quad (9.10)$$

а из (9.5) и (9.8) получаем

$$kt_{\tau} < \sum_{i=1}^k (t_i + t_{i+1}) = 2t_{\text{посл}}. \quad (9.11)$$

Из (9.10) и (9.11) получаем

$$k/2 < P_{\text{конв}}^{\text{ном}}/P_{\text{посл}} \leq k. \quad (9.12)$$

В действительности рост реальной производительности процессора окажется ниже из-за простоев (задержек) конвейера. В процедурах выполнения некоторых команд (например, команд пересылки данных) отдельные этапы общего рабочего цикла отсутствуют, и, следовательно, простаивают отдельные блоки конвейера. Для команды условного перехода по результату предыдущей операции выборка следующей команды должна быть

задержана (конвейер простаивает несколько тактов), пока не будет сформирован признак результата (формируется на более позднем этапе) предыдущей операции.

Если  $p_m$  — вероятность выборки команды, вызывающей задержку конвейера на  $m$  тактов ( $m=1, 2, \dots, k$ ), то действительная производительность конвейера

$$P_{\text{конв}}^{\text{д}} = P_{\text{конв}}^{\text{ном}} / \left( 1 + \sum_{m=1}^k P_m m \right). \quad (9.13)$$

*Асинхронный конвейер команд.* При большой зависимости продолжительностей выполнения процедур отдельных этапов от типа команды и вида операндов целесообразно применение асинхронного конвейера, в котором отсутствует единый такт работы его блоков, а информация с одного блока конвейера передается на следующий, когда данный блок закончит свою процедуру, а следующий полностью освободится от обработки предыдущей команды.

Управление передачей информации между соседними блоками в асинхронном конвейере осуществляется с помощью двух триггеров — готовности блока (сигнализирует о завершении операции в блоке) и освобождения последующего блока.

В качестве примера применения асинхронного конвейера команд может служить процессор ЭВМ ЕС-1050, в котором реализован конвейер, выполняющий одновременно три команды [18]. Рабочий цикл выполнения команды разбит на три этапа: I — выборка очередной команды; II — формирование исполнительных адресов и выборка операндов; III — операция в АЛУ, формирование признака результата и запись результата в память.

Для каждого из указанных этапов выполнения команды имеется соответствующая аппаратура. Например, кроме сумматора АЛУ есть отдельный сумматор для формирования исполнительного адреса на этапе II. На рис. 9.37 представлена структура управляющего устройства с «жесткой» логикой процессора ЭВМ ЕС-1050, на которой показаны блоки, управляющие процедурами отдельных этапов выполнения команды.

На рис. 9.38 показана временная диаграмма совмещения выполнения трех команд в ЭВМ ЕС-1050. Временная диаграмма построена для случая, когда выбираемый за одно обращение к памяти «участок программы» содержит четыре команды формата RR.

Этап I содержит две процедуры: выборку из ОП участка программы (8 байт) и распаковку участка — выделение из него очередной команды и размещение ее в регистре команды.

Этап II в общем случае включает в себя формирование

Рис. 9.37. Структура управляющего устройства процессора:

БВК — блок выборки команд;  
БМП — блок местной памяти;  
БВД — блок выборки данных;  
БЦУ — блок центрального управления;  
БСА — блок сумматора адреса;  
БАР — блок адреса результата;  
АЛБ — арифметическо-логический блок; ПУ — пульт управления; УП — управляющие сигналы

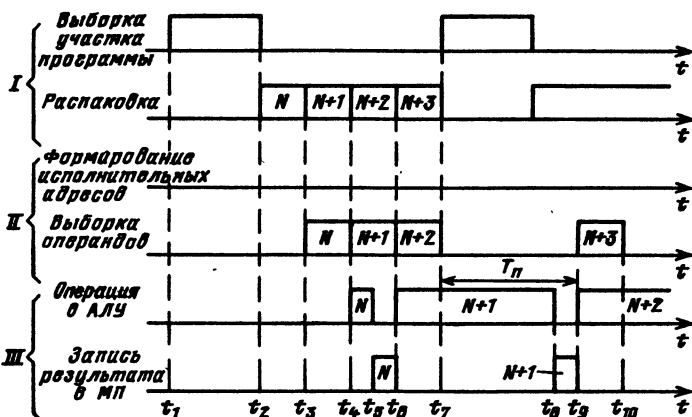
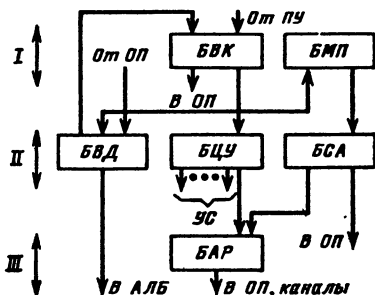


Рис. 9.38. Временная диаграмма совмещения выполнения трех команд (формата RR) в асинхронном конвейере команд (ЭВМ, ЕС-1050)

исполнительных адресов (при выполнении команд формата RR, отсутствует) и выборку операндов.

Этап III состоит также из двух процедур: выполнения операций в АЛУ и записи результата в память.

Из диаграммы видно, что, начиная с момента времени  $t_4$ , выполняются одновременно три этапа цикла соответственно для трех команд. В приведенном примере с момента  $t_7$  из-за большой длительности в команде  $N+1$  операции в АЛУ приостанавливается работа блоков аппаратуры, соответствующих этапам I и II.

**Арифметический конвейер.** Выше был рассмотрен конвейер команд. Однако в целях повышения производительности машины принцип конвейерной обработки широко используется и в самих выполняющих содержательную обработку информации устройствах (АЛУ), которые строятся в виде *арифметического конвейера*, причем таких арифметических конвейерных линий

может быть в процессоре несколько, в том числе и специализированных для определенных операций с данными. Подобные операционные (арифметические) устройства часто называют *магистральными*.

Пусть операционное устройство должно вычислять некоторую функцию  $\Phi$  от входных данных (выполнять некоторую операцию над входными данными). Можно функцию  $\Phi$  представить в виде последовательности более простых подфункций

$$\Phi_1 \rightarrow \Phi_2 \rightarrow \Phi_3 \rightarrow \dots \rightarrow \Phi_k,$$

причем такой, что результаты преобразования, выполняемые подфункцией  $\Phi_i$ , используются в качестве входных данных при вычислении подфункции  $\Phi_{i+1}$ , и если при этом для каждой подфункции иметь реализующую ее схемный блок, то получим арифметический конвейер, который может быть выполнен как синхронный или как асинхронный.

Приведенные выше элементы теории синхронного конвейера команд остаются в силе и для синхронного арифметического конвейера. Если  $t_t$  — такт конвейера, то после полной загрузки он станет выдавать значения функции  $\Phi$  через интервалы времени  $t_t$ . Увеличение производительности процессора за счет использования арифметического конвейера можно оценить по (9.12).

Если арифметический конвейер используется для выполнения разных операций, то осложняется определение состава рабочих позиций (блоков) конвейера и может потребоваться настройка (диспетчеризация) с соответствующей коммутацией блоков конвейера на операцию, задаваемую текущей командой.

Рассмотрим в качестве примера использование арифметического конвейера для сложения двух векторов  $X + Y = Z$ , компонентами которых являются числа, представленные в форме с плавающей точкой и в нормализованном виде.

Выделим в операции сложения чисел с плавающей точкой четыре этапа: 1) сравнение и определение разности порядков; 2) выравнивание порядков — сдвиг мантиссы числа с меньшим порядком на число разрядов, равное разности порядков; 3) сложение мантисс; 4) нормализация результата.

В арифметическом конвейере эти этапы выполняются отдельными блоками, образующими конвейер, по которому перемещаются операнды или промежуточные результаты операции. По мере их перемещения в конвейер вводятся новые компоненты векторов.

Пусть времена, необходимые для выполнения этапов сложения чисел с плавающей точкой, есть  $t_1, t_2, t_3, t_4$ . Следовательно,

Рис. 9.39. Пример настройки арифметического конвейера на выполнение различных операций

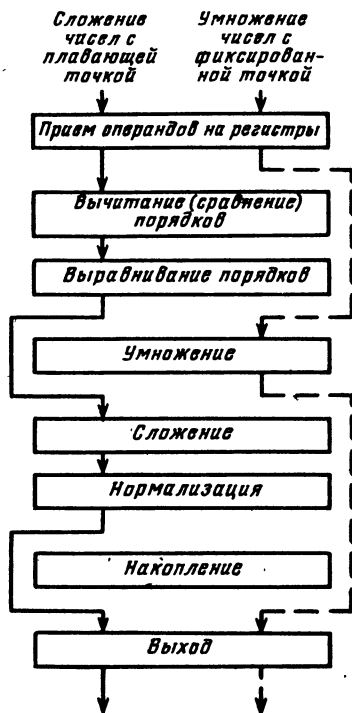
если не организовать конвейер и выполнять все этапы операции последовательно, то для получения компонента  $z_i = x_i + y_i$  потребуется время  $T = t_1 + t_2 + t_3 + t_4$ .

В синхронном конвейере, как указывалось выше, продолжительность каждого этапа устанавливается по самому длинному из них, пусть в данном случае это  $t_3$ . Тогда, если конвейер заполнен, результаты сложения элементов векторов будут выдаваться через каждые промежутки времени  $t_3$ , т. е. значительно быстрее, чем в случае отсутствия конвейерной обработки.

На рис. 9.39 в качестве примера представлена структура конвейерного (магистрального) АЛУ, соответствующего АЛУ известной в свое время ЭВМ ASC фирмы Texas Instruments, и показаны варианты коммутации блоков конвейера для выполнения разных операций, в данном случае сложения чисел с плавающей точкой и умножения чисел с фиксированной точкой.

Особенно эффективно использование операционных (арифметических) конвейеров в специализированных вычислительных устройствах с ограниченным набором алгоритмов обработки входных потоков данных, так как в этом случае возможно разбиение АЛУ на большое число простейших быстродействующих конвейерных блоков при небольших схемных и временных потерях на их коммутацию.

В ряде процессоров ЭВМ и микропроцессоров одновременно присутствуют конвейер команд и арифметический конвейер, при этом часто в процессоре (микропроцессоре) выделяют *I*-часть — аппаратуру, относящуюся к обработке собственно команд и *E*-часть — аппаратуру, связанную с операциями над данными<sup>1</sup>.



<sup>1</sup> *I* — от Instruction (инструкция, команда) и *E* — от Execution (выполнение).

## Контрольные вопросы

1. Что относится к элементам архитектуры ЭВМ?
2. Что определяет остроту проблемы при выборе структуры и формата команд современных ЭВМ? Каковы пути решения этой проблемы?
3. Что такое самоопределяемые данные? Почему при использовании тегов сокращается количество различных команд в системе команд машины?
4. Почему в малоразрядных ЭВМ и микропроцессорах широко используется косвенная адресация? Приведите пример совместного использования регистровой и косвенной адресации.
5. Поясните, почему стековая память позволяет использовать безадресные команды?
6. Каковы назначение и особенности реализации команды безусловного перехода с возвратом?
7. Как с помощью индексации организуется обработка упорядоченных массивов данных?
8. Каковы назначение и процедуры автоинкрементной и автодекрементной адресаций?
9. Укажите, в каком из изображенных на рис. 9.17 форматов команд ЕС ЭВМ используется косвенная адресация?
10. Что общего между вектором состояния программы (процессора) и вектором прерывания?
11. Каковы назначение и процедура прерывания программ ЭВМ?
12. Что такое векторное прерывание? Опишите процедуру векторного прерывания с использованием стековой памяти.
13. Почему реализуемые в архитектуре ЕС ЭВМ программные прерывания и прерывания при обращении к супервизору не могут возникать одновременно?
14. В чем различие синхронного и асинхронного конвейеров?
15. Каким образом особенности архитектуры ЭВМ с сокращенным набором команд способствуют повышению ее быстродействия? Какова при этом роль «перекрывающихся регистровых окон»?

## Глава 10

# ОРГАНИЗАЦИЯ МИКРОПРОЦЕССОРОВ И ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

## 10.1. Общие сведения о микропроцессорах

Развитие интегральной технологии и схемотехники цифровых электронных схем привело к появлению интегральных микросхем с большой и сверхбольшой степенями интеграции (БИС

и СБИС), содержащих на одном кристалле (в одном корпусе) несколько десятков тысяч, а в последних разработках сотни тысяч элементарных транзисторов. На основе таких схем в последние годы удалось создать микропроцессоры — функционально законченные, управляемые хранимой в памяти программой (большей частью малоразрядные) устройства обработки цифровой информации, выполненные в виде одной или нескольких БИС или СБИС.

*Микропроцессоры* (МП) отличаются крайне малыми габаритными размерами, малой потребляемой мощностью, довольно большим быстродействием, высокой надежностью и дешевизной. Очень важной особенностью микропроцессоров является их универсальность, т. е. возможность самого разнообразного применения благодаря их программируемости на выполнение конкретных функций.

Малые габаритные размеры, универсальность, способность реализовать сложные функции обработки данных и управления служат основой для построения различных микроЭВМ, персональных компьютеров, микропроцессорных устройств и систем управления, встраиваемых в различные аппараты, машины, орудия, приборы и системы.

*Микропроцессорные средства* производятся в виде микропроцессорных комплектов интегральных микросхем, имеющих единое конструктивно-технологическое исполнение и предназначенных для совместного применения. Микропроцессорный комплект помимо самого микропроцессора содержит микросхемы, поддерживающие функционирование микропроцессора и расширяющие его логические возможности.

Развитие микропроцессоров ознаменовалось соревнованием биполярной и МОП-технологий микроэлектроники.

МОП-структуры электронных схем позволяют размещать на одном кристалле большое число элементарных схем благодаря их небольшому размеру и небольшой мощности рассеяния. Однако первые МОП-микропроцессоры имели сравнительно низкое быстродействие (сложение двух слов в регистрах занимало около 20 мкс).

Биполярные БИС, например ТТЛ-схемы, обладали намного большим быстродействием, но значительно меньшей плотностью компонентов на кристалле. Поэтому довольно трудно было построить биполярный микропроцессор на одном кристалле.

Для преодоления ограничений, связанных со сравнительно небольшой плотностью компонентов у биполярных схем, был предложен *секционный метод* конструирования микропроцессоров.



По этому методу микропроцессор составляется из нескольких одинаковых 2-, 4- или 8-разрядных процессорных секций, размещенных на отдельных кристаллах и объединенных общим микропрограммным управлением. Использование секционного метода в сочетании с прогрессивной технологией маломощных ТТЛ-схем с диодами Шотки (ТТЛШ) позволило создать микропроцессорные комплекты серий К589, КР1802, КР1804, содержащие соответственно 2-, 4-, 8-разрядные процессорные кристаллы (секции) с быстродействием (временем цикла) 100—150 нс. В состав этих комплексов помимо БИС процессорной секции входят пять—семь дополнительных БИС (блоки микропрограммного управления, синхронизации, связи с периферийными устройствами, прерываний). К рассматриваемому типу микропроцессорных средств относится реализованная на ЭСЛ-технологии серия быстродействующих микропроцессорных БИС К1800, содержащая 4-разрядные секции АЛУ, микропрограммного управления, синхронизатора, управления оперативной памятью.

На основе секционных микропроцессорных БИС можно строить микропроцессоры с изменяемой разрядностью слова и микропрограммным управлением. Эти микропроцессоры не имеют фиксированного набора команд, а имеют только набор микрокоманд, что позволяет (хотя это связано с определенными трудностями для широкого пользователя) реализовать микропрограммным путем оптимальный для данной задачи набор команд и отдельных процедур.

Секционные ТТЛШ и ЭСЛ-микропроцессорные серии БИС используются при построении устройств вычислительной техники и автоматики, которым предъявляются повышенные требования в отношении быстродействия.

Параллельно с этим направлением быстро совершенствовалась МОП-технология и были достигнуты крупные успехи в повышении степени интеграции и быстродействия. На этой основе по мере совершенствования МОП-технологии были созданы 8-, а затем 16-разрядные с жесткой логикой и фиксированной системой команд, а в последнее время — 32-разрядные однокристалльные микропроцессоры.

В настоящее время в микропроцессорной технике основное место занимают однокристалльные микропроцессоры, выполненные на различных вариантах МОП-технологии. Они служат основой современных микроЭВМ и персональных компьютеров, широко используются в различных микропроцессорных устройствах и системах автоматизации управления и обработки данных.

## 10.2. Организация однокристальных 8-разрядных микропроцессоров

Восьмиразрядный микропроцессор с фиксированной системой команд К580ВМ80А, в дальнейшем для краткости обозначаемый МП К580, выполнен в виде изготовленной по *n*-МОП-технологии БИС, содержащей около 5 тыс. транзисторов на кристалле кремния размером около 30 мм<sup>2</sup>, заключенном в корпусе с 40 выводами. Работает с тактовой частотой 2 МГц (длительность такта 500 нс), требует трех уровней напряжения питания: +5, —5, +12 В. Микропроцессор К580 аналогичен МП 8080 фирмы Intel (США).

Длина машинного слова в МП К580 8, а адреса 16 разрядов. Микропроцессор может работать с оперативной и постоянной памятью, суммарная емкость которых не превышает 64 Кбайт. Ширина выборки из памяти 1 байт. Возможна адресация любого байта памяти. Скорость выполнения коротких операций (типа регистр — регистр) 500 тыс. операций/с.

Создание высокопроизводительного микропроцессора с эффективными системой команд и процедурами обмена информацией с внешним по отношению к микропроцессору оборудованием затруднялось из-за ограничений, которые вызывались коротким словом МП и малым числом внешних выводов его корпуса. Последнее, в частности, обусловило узкую (малоразрядную) шину данных в МП и узкий интерфейс обмена данными с внешним оборудованием.

Для преодоления этих ограничений потребовались разработки ряда новых архитектурных решений. С частью из них читатель познакомился в гл. 9. Здесь остановимся главным образом на структурных аспектах и особенностях системы команд МП. Организация ввода-вывода в МП рассмотрена в гл. 11.

Читатель имел возможность ознакомиться с программистской моделью МП К580, приведенной в гл. 9. Более детальное представление об устройстве МП К580 дает его структура, приведенная на рис. 10.1.

Характерными особенностями организации МП К580 являются:

- трехшинная структура с шинами данных, адреса и управления;

- магистральная структура связей — наличие внутренней шины данных, которая связывает все узлы внутри МП. Ширина шины данных (8 разрядов) равна разрядности слов, с которыми оперирует МП;

- регистровая память в виде блока программно-доступных

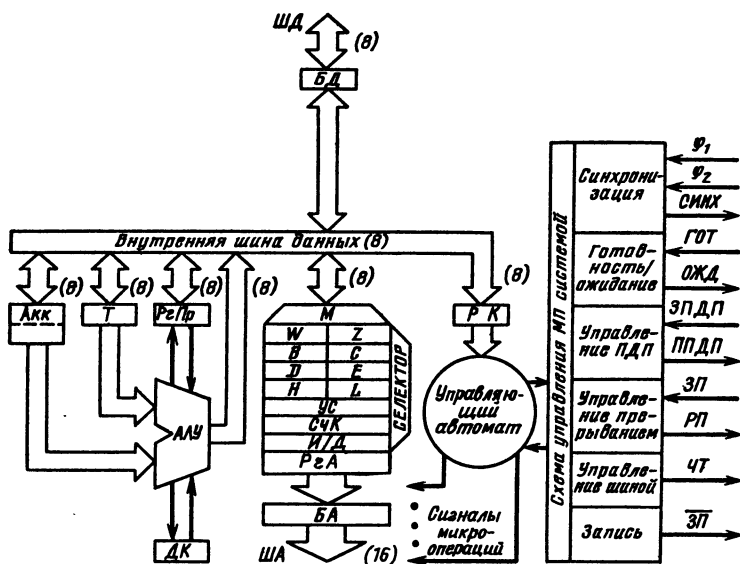


Рис. 10.1. Структура однокристального 8-разрядного МП К580:  
 Узлы: Акк — аккумулятор; Т, ш, z — регистры временного хранения; РзПр — регистр признаков; М — мультиплексор; И/Д — инкрементор/декрементор; РзА — регистр адреса; ДК — десятичный корректор; РзК — регистр команд; СчК — счетчик команд; УС — указатель стека; БД — буфер данных; БА — буфер адреса; шины: ША — шина адреса; ШД — двунаправленная шина данных; сигналы: ЗП — запись; ЧТ — чтение; ЗП — запрос прерывания; ЗПДП и ППДП — запрос и подтверждение прямого доступа к памяти; ОЖД — ожидание; ГОТ — готовность; СИНХ — синхронизация;  $\varphi_1$  и  $\varphi_2$  — тактовые сигналы; РП — разрешение прерывания

общих и некоторых специализированных регистров (СчК, указатель стека, регистр косвенного адреса, регистр признаков);

наличие 16-разрядной шины адреса, позволяющей прямо адресовать внешнюю память емкостью 64 Кбайт;

разнообразие применяемых методов адресации — прямая, регистровая прямая, регистровая косвенная (в том числе подразумеваемая), непосредственная, индексная, стековая, — в совокупности позволяющих при коротком 8-разрядном слове реализовывать достаточно гибкую систему команд, использующую три формата команд: 1, 2, 3 байта;

наличие эффективных средств работы с подпрограммами и быстродействующей системой прерывания, использующих стековую память («перевернутый стек»), специальные команды вызова подпрограмм и возврата (в том числе условного) из подпрограмм и процедуры перехода к прерывающим и возврата к прерванным программам;

реализация двухбайтных (тандемных) передач и некоторых других двухбайтных операций, с тем чтобы при 8-разрядных шине данных, общих регистрах и ширине выборки из ОП упростить процедуры обработки 16-разрядных слов и работу с 16-разрядными адресами и тремя форматами команд. Последнее достигается тем, что первый байт команды, содержащий указание об ее формате, загружается всегда в регистр команды, а второй и третий, если они имеются, — в определенные регистры.

Помимо упоминавшихся шин данных и адреса имеется шина управления, содержащая линии, предназначенные для передачи управляющих сигналов, признаков состояния процессора и периферийного оборудования. Шина содержит следующие линии: синхронизирующих сигналов для сопровождения информации при передачах ее в обоих направлениях по мультиплексируемой шине данных;

сигналов, информирующих МП о состоянии (готовности) периферийных устройств;

сигналов запроса прерывания от периферийных устройств и разрешения прерывания и др.

Блок регистров содержит программно-доступные (с регистровой прямой или подразумеваемой адресацией) регистры: 8-разрядные аккумулятор *A*, общие регистры *B*, *C*, *D*, *E*, 16-разрядные счетчик команд *CчК*, указатель стека *УС*, парный регистр косвенного адреса *H — L*, а также 8-разрядный регистр признаков, отдельные разряды (флажки) которого, принимая значение 1, указывают: *СУ* — перенос, *P* и *M* — соответственно знак + и — результата, *АС* — вспомогательный перенос, *Z* — нулевой результат. Кроме того, имеется регистр адреса *PzA*. Регистры используются для хранения операндов, промежуточных результатов и адресов.

Непосредственно в МП из оборудования стековой памяти содержатся только указатель стека и соответствующие цепи управления. Сам стек реализуется в виде зоны ячеек в общей ОП. Хотя такая организация стека по сравнению с внутренним (встроенным в кристалл МП) стеком связана с некоторым замедлением стековых процедур, однако операции со стеком выполняются намного реже, чем с другими регистрами. При таком выполнении стека его емкость практически неограниченна, оказывается возможным большее число вложений при работе с подпрограммами и прерываниями, что в общем обеспечивает высокую производительность МП.

Блок регистров содержит специальную схему «инкрементор»/«декрементор» И/Д, позволяющую без привлечения АЛУ в процессе межрегистровых передач выполнять модификацию текущего адреса добавлением (вычитанием) 1.

Арифметическо-логическое устройство, выполненное в виде комбинационной схемы с собственным регистром временного хранения, производит арифметические и логические операции над 8-разрядными операндами. Один вход АЛУ связан с аккумулятором, а другой может быть соединен с любым общим регистром. Имеется включаемый специальной командой десятичный корректор, превращающий результат непосредственно предшествующей операции двоичного сложения (вычитания) над двоично-десятичными операндами в результате соответствующей операции десятичной арифметики (см. гл. 7).

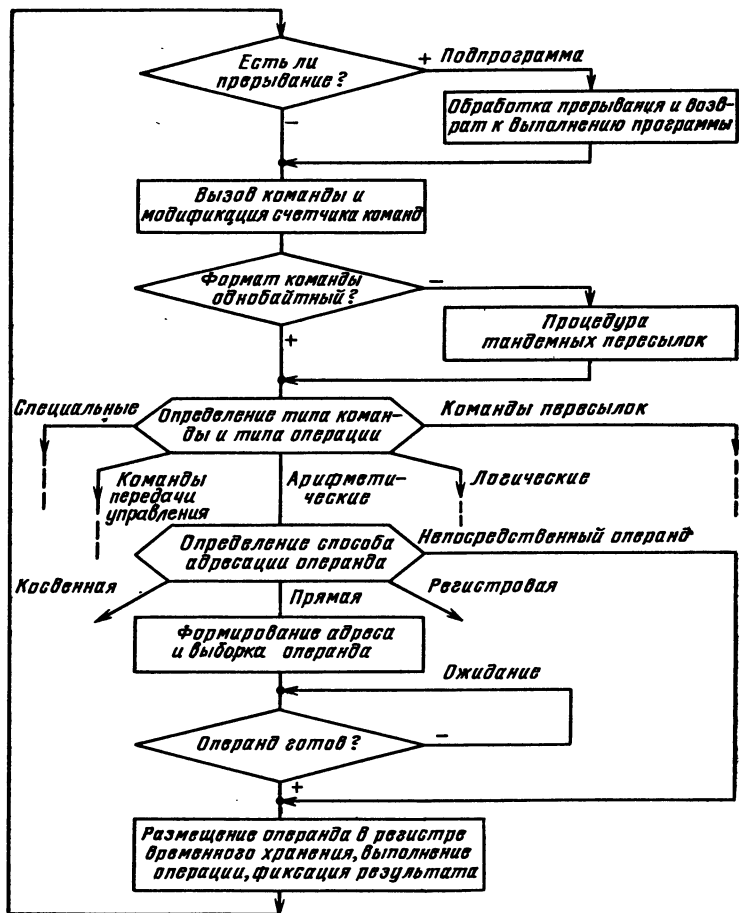


Рис. 10.2. Схема рабочего цикла 8-разрядного микропроцессора

По завершении арифметических и логических операций 1 в соответствующих разрядах (флажках) регистра признаков указывают на наличие переполнения, положительного, отрицательного или нулевого результата.

Передача управления в командах условного перехода происходит в зависимости от состояния определенных разрядов (флажков) регистра признаков.

В МП применены буферные схемы шины данных и адреса. Буферная схема состоит из регистра и выходных схем с тремя состояниями: 0, 1 и полное электрическое отключение от нагрузки («высокоомпеданское» или «плавающее» состояние). Наличие третьего состояния позволяет реализовывать магистральный принцип связи между отдельными узлами внутри МП и между модулями (ОП, периферийные устройства), присоединяемыми наряду с МП к общему интерфейсу «мультишина» (см. гл. 11) микропроцессорной системы (или микроЭВМ).

Управляющий автомат с «жесткой» логикой, реализованный с использованием программируемой логической матрицы, выдает последовательность управляющих сигналов, инициирующих процедуру выполнения текущей команды. Последовательность сигналов и сама процедура зависят от задаваемых кодом команды формата команды и операндов, способа адресации.

На рис. 10.2 показана схема рабочего цикла выполнения команды в микропроцессоре, которая приводится, главным образом, с целью показать читателю, что в основных чертах она не отличается от схем рабочего цикла крупных ЭВМ.

Микропроцессорный комплект интегральных микросхем серии К580 помимо самого микропроцессора содержит следующие микросхемы (кристаллы): контроллер прямого доступа к памяти, программируемый таймер, последовательный и параллельный интерфейсы, контроллер прерываний и др.

### ***Особенности системы команд МП К580***

Микропроцессор К580 выполняет арифметические операции над 8-разрядными, а с уменьшенным быстродействием — и над 16-разрядными словами, которые могут быть целыми двоичными числами со знаком и без знака, десятичными двоично-кодированными числами, и логические операции над цепочками данных.

Система команд микропроцессоров К580 выглядит довольно упрощенной по сравнению с системами команд ЭВМ, что объясняется коротким 8-разрядным машинным словом и ограниченными аппаратными ресурсами. Операции умножения и деления над целыми числами, операции с плавающей точкой выполняются по подпрограммам или с помощью дополнительного

специализированного арифметического сопроцессора («арифметического расширителя»).

Для упрощения организации и убыстрения операций с 16-разрядными адресами и операндами в систему команд введены команды 2-байтных («тандемных») передач, арифметических и других операций, в которых некоторые пары регистров адресуются как один регистр, а их содержимое рассматривается как одно 16-разрядное слово, передаваемое по шине данных тандемом — последовательно байт за байтом. В 8-разрядном АЛУ арифметические операции над 16-разрядными числами выполняются двумя командами: первая задает операцию над младшими, а вторая — над старшими байтами. Эти вторые команды — специфические, они задают операцию сложения (вычитания) байт, в которых участвует перенос (заем), полученный в операции с младшими байтами и запомненный в триггере переноса. Имеется также несколько команд, непосредственно обрабатывающих 16-разрядные числа.

Микропроцессор К580 имеет три формата команд: однобайтный (однословный), двухбайтный и трехбайтный (рис. 10.3). Формат команды и тип адресации задаются неявно кодом операции. Адрес команды задается адресом ее первого байта.

Проблема построения системы команд при коротком машинном слове решается благодаря использованию регистра-аккумулятора с подразумеваемой адресацией для реализации одноадресных и безадресных команд. В последних адрес операнда неявно задается кодом операции. Широко применяются укороченная регистровая адресация для обращения к общим регистрам и регистровая косвенная и индексная адресации для задания операнда в ОП. Наличие в регистровой структуре специального 16-разрядного регистра косвенного адреса позволяет иметь команды с подразумеваемой косвенной адресацией, т. е. без указания в команде регистра, хранящего исполнительный адрес.

В двухбайтных командах реализуется непосредственная адресация, а в трехбайтных — прямая адресация ячейки памяти.

Схема «инкрементатор»/«декрементатор» позволяет реализовать процедуры автоматического задания приращений не только в указателе стека, но и в счетчике команд, в индексном регистре, в регистре косвенного адреса и т. д. Операции *инкремент* и *декремент* выполняются в процессе межрегистровых передач.

Команды с прямой адресацией занимают много места в памяти и затрачивают много времени на выполнение из-за многократных обращений к памяти. Поэтому по возможности следует стремиться к использованию команд с регистровой, непосред-

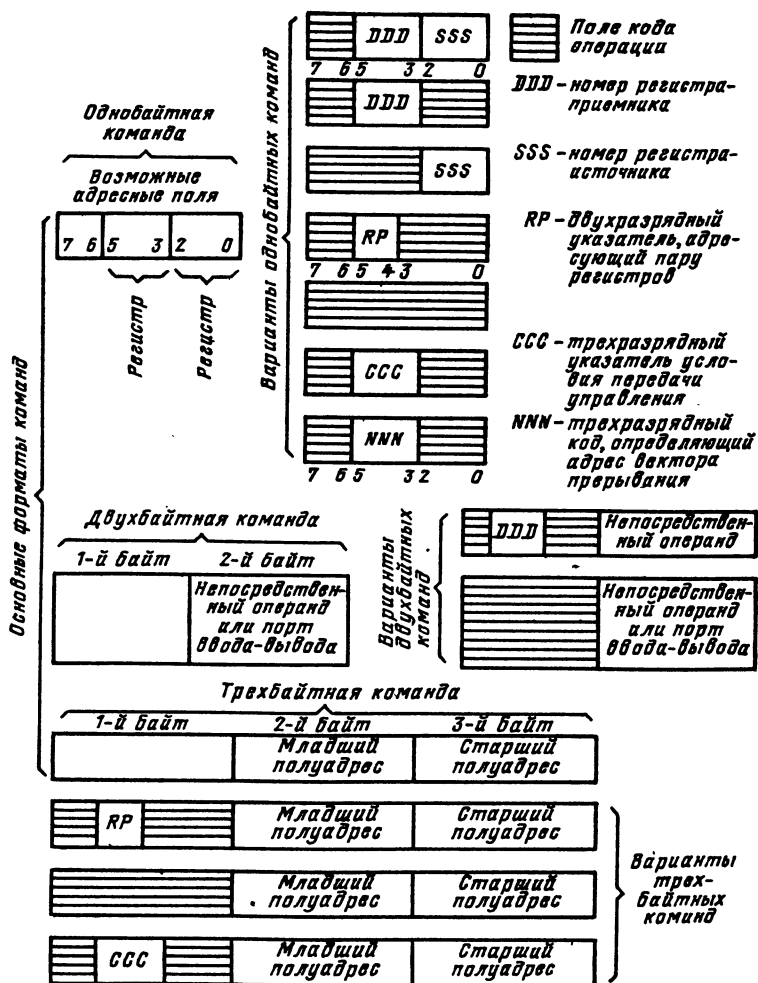


Рис. 10.3. Форматы команд МП К580

ственной и регистровой косвенной адресациями. При этом надо помнить, что регистровая косвенная адресация требует предварительной настройки, т. е. загрузки адреса в регистр косвенного адреса. Поэтому регистровая косвенная адресация оказывается эффективной при обработке списочных структур данных, когда после загрузки косвенного адреса он многократно модифицируется командами *инкремент* или *декремент*.



В командах условного перехода 3-разрядный код ССС задает в трехбайтной команде условие передачи управления по адресу, указанному в команде. Возможны задания восьми вариантов условия перехода: по переполнению, отсутствию переполнения, нулевому, ненулевому, положительному, отрицательному, четному и нечетному результатам.

Для упрощения перехода к подпрограмме и возврата из нее используются стековая память (в данном случае «перевернутый стек») и ряд специальных команд.

Имеются команды вызова подпрограммы и возврата из подпрограммы. По первой из них (трехбайтной) содержимое *СчК* побайтно загружается в стек (указатель стека перед загрузкой каждого байта уменьшается на 1), а указанный в команде адрес начала подпрограммы передается в *СчК*. По второй (однобайтной) команде, которая ставится в конце подпрограммы, из стека в *СчК* побайтно передается двухбайтный адрес возврата в основную программу (после передачи каждого байта УС увеличивается на 1). Дополнительную гибкость в организации работы с подпрограммами обеспечивают оригинальные команды *вызова по условию подпрограммы и возврата по условию из нее*. Условия те же, что и у команды условного перехода.

Для управления стеком имеются команды передачи из заданной команды пары регистров в стек и обратно, из регистра признаков в стек и обратно, команды обмена содержимого стека и регистровой пары, загрузки указателя стека.

Всего имеется две команды ввода-вывода: *ввести* и *вывести* (двухбайтный формат). В команде во втором байте указывается номер порта ввода-вывода, из которого байт информации соответственно вводится или куда байт выводится.

### ***Система прерывания 8-разрядного микропроцессора К580***

Система прерывания имеет много общего с рассмотренной в гл. 9 системой прерывания малых ЭВМ, что определяется в первую очередь тем, что в этом случае используются модификация интерфейса типа «общая шина» («мультишина»), порог прерывания и стековая память. Есть и отличия, например, в формировании адреса и структуре вектора прерывания.

Структура вектора состояния (а следовательно, и вектора прерывания) представлена на рис. 9.21. Он занимает четыре 8-разрядных слова. Векторы прерываний для восьми уровней прерывания хранятся в фиксированных ячейках начальной области памяти с адресами 0, 8, 16, ..., 56.

Рассмотрим упрощенный вариант системы прерывания на основе использования микросхемы «блок приоритетного преры-

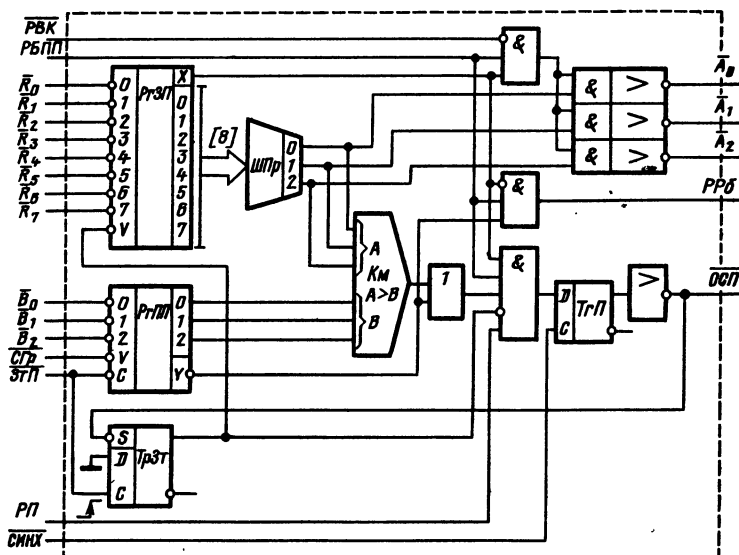


Рис. 10.4. Схема блока приоритетного прерывания:

узлы:  $PзП$  — регистр запросов прерывания;  $PзПП$  — регистр порога прерывания;  $ТрЗт$  — триггер запрета прерывания;  $ШПр$  — шифратор приоритета запроса прерывания;  $Км$  — компаратор;  $ТзП$  — триггер прерывания; сигналы:  $PВК$  — разрешение выдачи кода приоритета;  $PБПП$  — разрешение работы данного БПП в многоблочной системе прерываний;  $СГр$  — выбор состояния в группе [подаётся от дешифратора в многоуровневых (более восьми) системах прерывания];  $ЗтП$  — запрет прерывания;  $PП$  — разрешение прерывания;  $ОСП$  — общий сигнал прерывания;  $СИНХ$  — сигнал синхронизации триггера прерывания;  $PP6$  — сигнал разрешения работы следующему (в сторону уменьшения приоритета) БПП в многоблочной системе прерывания

вания (БПП)», схема которого представлена на рис. 10.4<sup>1</sup>. Этот блок реализует восьмиуровневую систему прерывания. Запросы поступают на входы  $R_7—R_0$  (приоритет растёт с увеличением номера входа). Число уровней входов может быть увеличено присоединением дополнительного БПП. При одновременном появлении нескольких запросов (запрос соответствует низкому уровню потенциала) в регистр запросов прерывания будет записана 1 лишь в разряд, соответствующий наиболее приоритетному из поступивших запросов, так как сигнал запроса  $R_i=0$  на одном из входов блокирует все входы с меньшими номерами.

<sup>1</sup> Более широкими возможностями в отношении организации ввода-вывода по прерываниям обладает входящая в состав микропроцессорного комплекта серии K580 микросхема «Программируемый контроллер прерываний» K580BH59 [25].

Восьмиразрядный унитарный код запроса поступает в шифратор, формирующий 3-разрядный двоичный код номера запроса  $N = A_0 A_1 A_2 = A$ , который в компараторе  $K_m$  сравнивается с кодом порога прерывания (приоритета текущей программы)  $B = B_0 B_1 B_2$ , хранящимся в регистре порога прерывания (регистре состояния). Если уровень запроса выше установленного идущей на процессоре программой порога прерывания ( $A > B$ ), то на выход компаратора поступает сигнал 1, в противном случае 0.

Блок приоритетного прерывания находится в режиме ожидания сигнала разрешения стробирования (на схеме не показан), который формирует микропроцессор по окончании цикла команды. Если в момент стробирования БПП, т. е. в момент поступления от микропроцессора сигнала разрешения прерывания (РП) ( $INTE$ ), на выходе компаратора единичный сигнал, то триггер  $T_2П$  переходит в состояние 1 и на выходе буферной схемы формируется общий сигнал прерывания  $\overline{ОСП}$  ( $\overline{INT}$ ), который устанавливает триггер запрета прерывания  $Tr_3T$ , кратковременно блокирующий регистр  $P_2ЗП$  на случай появления запросов с более высоким приоритетом. Сигнал  $\overline{ОСП}$  ( $\overline{INT}$ ) инициирует формирование кода команды передачи управления

$RST\ N$ ,

которая в данном случае имеет вид 11  $A_0 A_1 A_2$  111, при этом 3-разрядный номер  $N = A_0 A_1 A_2$  принятого к обслуживанию запроса передается в микропроцессор по сигналу  $\overline{PBK}$  ( $\overline{ELR}$ ). Протекание процедуры прерывания поясняет рис. 10.5.

По команде  $RST\ N$  аппаратура МП записывает в стек (в ОП по адресу из  $УС$ ) содержимое  $СчК$  (адрес возврата) и загружает в  $СчК$  адрес (соответствует числу  $8 \times N$ )

$$\underbrace{00 \dots 0 A_0 A_1 A_2 000}_{16 \text{ разрядов}},$$

по которому произойдет передача управления.

Код  $N = A_0 A_1 A_2$  определяет адрес вектора прерывания соответствующей прерывающей программы, а точнее, начальный адрес группы ячеек ОП, хранящей информацию о начальном адресе прерывающей программы. В рассматриваемом МП сохранение в памяти вектора состояния прерываемой программы и замена его в регистрах вектором прерывания прерывающей программы выполняются в основном не аппаратурными, а программными средствами. Поэтому в данном случае элементы вектора прерывания хранятся в ОП в виде операндов фрагмента

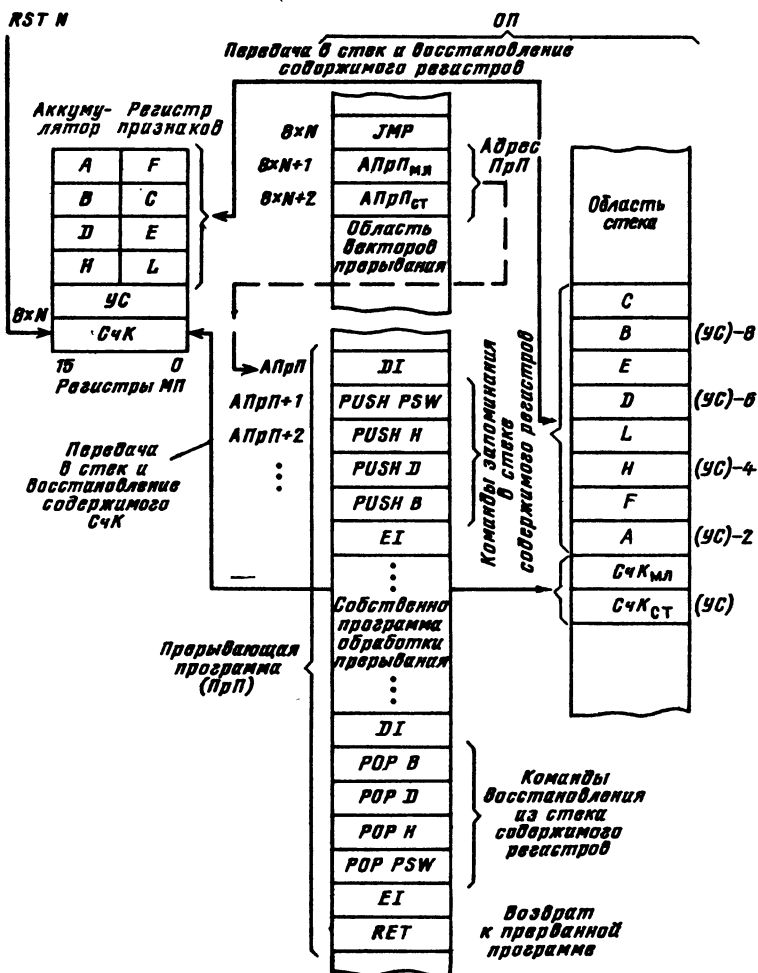


Рис. 10.5. Процедура прерывания программ в МП К580:

*JMP* — передача управления; *DI* и *EI* — запрещение и разрешение прерывания; *PUSH* и *POP* — запись в стек и считывание из стека; *RET* — загрузка из стека счетчика команд *СЧК* (возврат к прерванной программе)

программы, выполняющей указанные процедуры (рис. 10.5). Код *N* может не только формироваться аппаратурой, как это было описано выше, но и передаваться источником прерывания, например периферийным устройством, по шине данных интерфейса вместе с запросом прерывания (см. гл. 11).

Управление переходит к команде, загружающей в *СЧК* адрес

начала прерывающей программы. Начальная часть этой программы помещает в стек содержимое регистра-аккумулятора *A* и регистра признаков *F* (команда *PUSH PSW*), а также и других программно-доступных регистров МП, если прерывающая программа будет их использовать, при этом каждой командой передачи в стек передается содержимое соответствующей пары регистров. Далее выполняется собственно программа обработки прерывания.

Перед выходом из прерывающей программы последняя восстанавливает из стека состояния регистров, в том числе аккумулятора и регистра признаков, которые они имели перед прерыванием, и управление переходит к прерванной программе (команда *RET*). Перед выполнением процедуры запоминания и восстановления содержимого регистров прерывание запрещается (команда *DI*), а по их завершении разрешается (команда *EL*).

### 10.3. Организация однокристалльных 16-разрядных микропроцессоров

На основе *n*-МОП-технологии с кремниевыми затворами удалось реализовать принцип пропорционального уменьшения размеров МОП-схем, достигнуть большей степени интеграции (около 30 тыс. транзисторов на кристалле размером 5,5 × 5,5 мм) и высокого быстродействия. Задержка на элемент уменьшилась, она имеет тот же порядок, что и в ТТЛ-схемах с диодами Шотки, имеющими значительно большие размеры и потребляемую мощность.

На МОП БИС указанного типа фирмой Intel (США) был создан однокристалльный 16-разрядный МП 8086 (прототип отечественного МП КМ1810ВМ86, в дальнейшем для сокращения именуемого К1810), который по уровню производительности и логической организации не уступает средним моделям малых ЭВМ. Приборы выпускаются в 40-контактном корпусе.

Производительность МП К1810 составляет при тактовой частоте 5 МГц 2,5 млн. операций типа регистр-регистр. Это достигнуто благодаря повышению быстродействия схем и архитектурным усовершенствованиям. Архитектура МП К1810 имеет следующие особенности:

- выполнение аппаратурными средствами арифметических операций над 8- и 16-разрядными двоичными числами со знаком и без знака, десятичными двоично-кодированными числами, логических операций под цепочками данных, расширенные возможности работы с отдельными разрядами слов;

- наличие 16-разрядного АЛУ с аппаратурной реализацией умножения и деления;

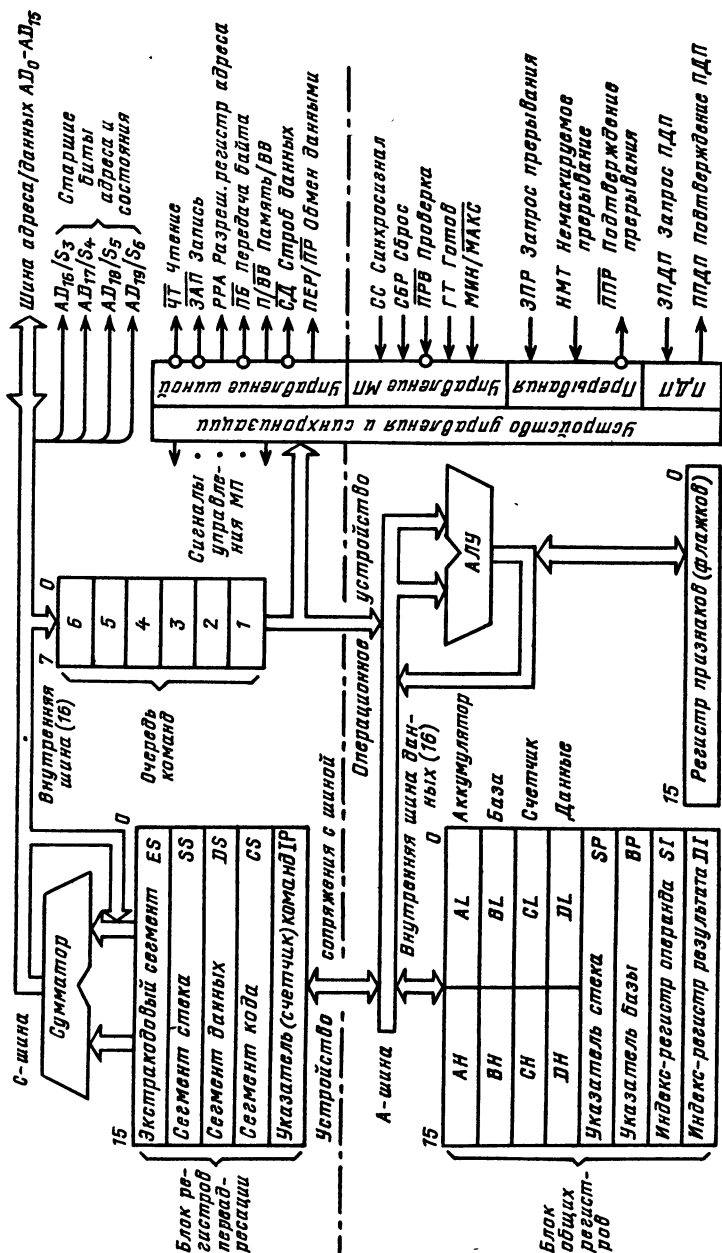


Рис. 10.6. Структурная схема 16-разрядного МП К1810

регистрация структура с удвоенным по сравнению с К580 числом общих регистров, практически неограниченное число уровней векторного прерывания;

полная совместимость по системе команд с МП К580 (в том числе работа с 8-разрядными командами последнего) и одновременно наличие новых, эффективных 16-разрядных команд;

сегментная адресация, позволяющая прямо адресовать од-номегабайтную память (оперативную, дисковую и т. п.), про-изводить динамическое перемещение программ;

использование одного уровня напряжения питания 5 В.

Связь между МП, ОП и периферийными устройствами осуществляется с помощью интерфейса ИА1 (*мультишина*) (см. гл. 11).

На рис. 10.6 представлена структурная схема МП К1810, в которой имеются относительно автономные устройства: а) *устройство сопряжения с шиной* (УСШ), обеспечивающее опережающую выборку команд и формирование очереди вы-бранных байт последовательности команд в специальной ре-гистровой памяти (емкость 6 байт), а также формирование физического адреса памяти, чтение операндов из памяти или регистров ввода-вывода и запись результата операции в память или регистры ввода-вывода; б) *операционное устройство* (ОУ), извлекающее команды из очереди и реализующее предписанные командами операции в 16-разрядном АЛУ.

Устройство сопряжения с шиной помимо регистров очереди команд имеет блок 16-разрядных регистров переадресации, 16-разрядный сумматор адреса. Сюда же можно отнести указа-тель (счетчик) команд.

Такая структура при определенном соотношении тактовой частоты МП и длительности цикла памяти позволяет получить эффективное совмещение процессов выборки и исполнения ко-манд. Одному циклу основной памяти (800 нс) соответствуют четыре такта работы МП К1810. При совмещении за время одного цикла основной памяти выполняются две однобайтные команды.

Операционное устройство включает в себя блок 16-разряд-ных общих регистров, содержащий четыре регистра данных: аккумулятор *АХ*, базовый регистр *ВХ*, «счетчик» *СХ* и «данные» *ДХ*, регистры — указатели стека *SP* и базы *BP* и индексные регистры операнда *SI* и результата *DI*, а также АЛУ и 16-раз-рядный регистр признаков (флажков) *F*.

Устройство управления и синхронизации управляет УСШ и ОУ, а также обменом данными с периферийными устройства-ми, включая обработку запросов прерывания и реализацию прямого доступа к памяти.

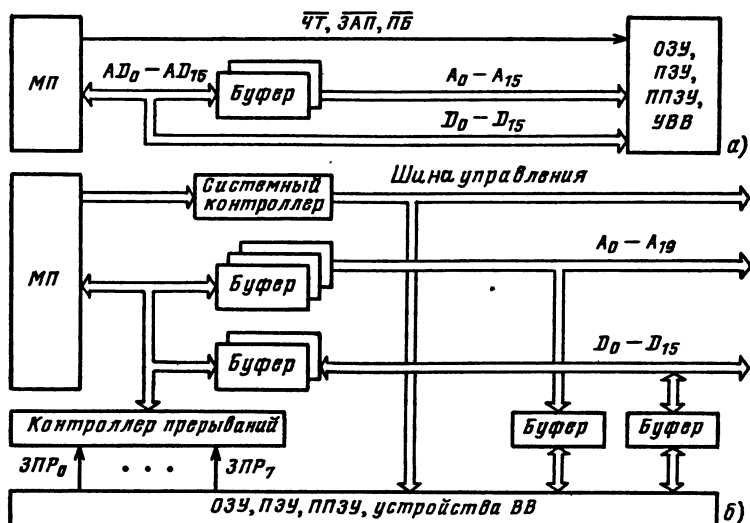


Рис. 10.7. Конфигурации МП-систем:

а — минимальная; б — максимальная

Микропроцессор K1810 имеет специальные аппаратные средства (сегментная адресация и др.), поддерживающие мультипрограммный режим работы и облегчающие создание на основе этого МП многопроцессорных систем, в том числе содержащих арифметические сопроцессоры, позволяющие в несколько раз увеличить скорость выполнения арифметических операций с плавающей точкой и вычисление некоторых функций.

Микропроцессор может работать в минимальной или максимальной конфигурации (рис. 10.7). Соответствующий режим работы устанавливается управляющим сигналом МИН/МАКС. При этом в максимальной конфигурации используется дополнительная микросхема — системный контроллер, вырабатывающий сигналы, управляющие записью и считыванием информации в памяти и в периферийных устройствах, и сигналы, подтверждающие прерывание. В минимальной конфигурации управляющие сигналы для памяти и устройств ввода-вывода вырабатывает сам МП K1810.

На рис. 10.8 представлена программистская модель МП K1810, содержащая его программно-доступные регистры [25, 35].

Регистры данных  $AX, BX, CX, DX$  служат для хранения операндов и результатов операций. Возможна адресация как целых регистров, так и их младшей  $L$  и старшей  $H$  частей. Не-



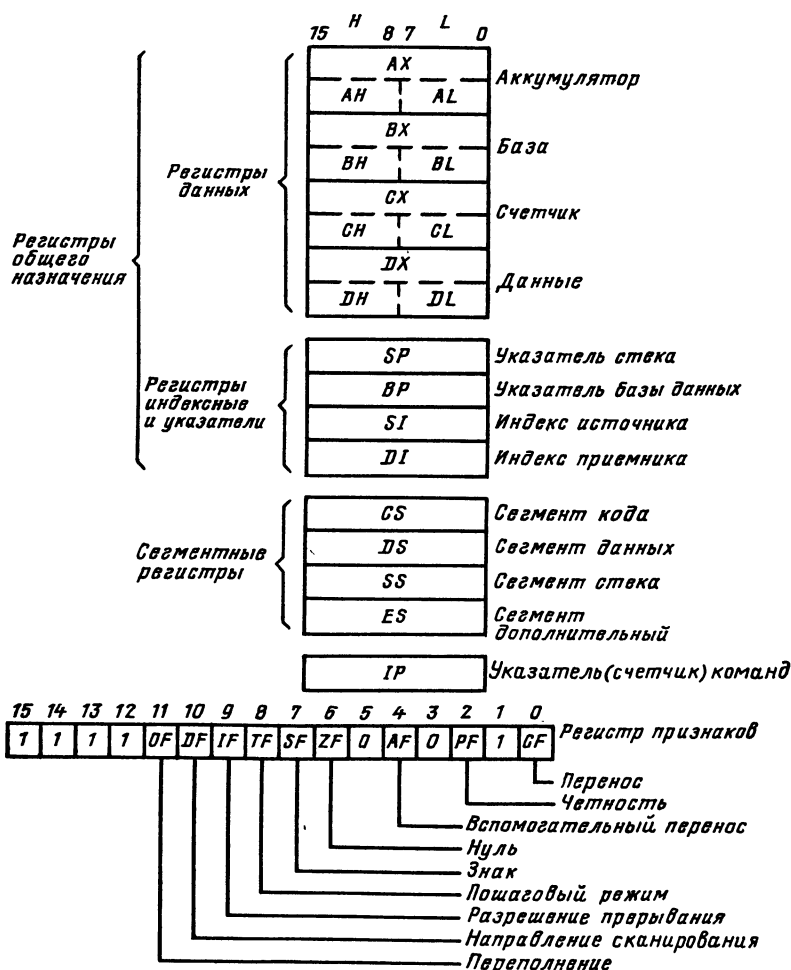


Рис. 10.8. Программистская модель МП К1810

которым регистрам наряду с общим придается и специальное назначение. В последнем случае возможно в соответствующих командах адресовать эти регистры неявно самим кодом операции (подразумеваемая адресация). Так, регистр *AX* используется в качестве аккумулятора, регистр *BX* — как базовый регистр, *CX* — как счетчик в командах сдвигов, управления вычислительными циклами и в операциях с цепочками байт, а регистр *DX* не-

явно адресуется в командах умножения и деления, а в некоторых операциях ввода-вывода хранит адрес порта ввода-вывода.

В МП К1810 используется сегментация памяти, организуемая с помощью сегментных регистров: кода *CS*, данных *DS*, стека *SS* и экстракода (дополнительного сегмента) *ES*, хранящих базовые адреса сегментов текущей программы. Эти базовые адреса должны быть кратны 16. Размер сегментов 64 Кбайт. Допускается перекрытие сегментов.

Группу указательных и индексных регистров, задающих внутрисегментные смещения, образуют регистры указателя стека *SP*, указателя базы стека (данных) *BP*, индекса операнда (источника) *SI* и индекса результата (приемника) *DI*. К этой группе можно отнести и регистр указателя (счетчика) команд *IP*.

Регистр признаков (флажков) *F*, который правильнее называть регистром состояния микропроцессора, имеет 16 разрядов, причем младшие 8 разрядов соответствуют регистру признаков (флажков) МП К580. В регистре признаков формируются: а) *признаки результата*: переполнения *OF* (при операциях с целыми числами); знака результата *SF*; нуля *ZF*; вспомогательного переноса *AF* (перенос из третьего или заем в третий разряд); четности *PF* (четное число единиц в младшем байте результата); переноса *CF* (перенос из старшего или заем в старший разряд результата); б) *признаки управления*: пошагового режима *TF* (управляет пошаговыми прерываниями); разрешения прерывания *IF* (разрешает или запрещает маскируемые прерывания); направления *DF* (указывает направления обработки цепочки данных, начиная с элемента с наименьшим адресом при  $DF=0$  или с наибольшим адресом при  $DF=1$ ).

Четыре 16-разрядных указательных и индексных регистров (*SP*, *BP*, *SI*, *DI*) могут участвовать в выполнении арифметических и логических операций над двухбайтными словами.

Память имеет байтовую организацию — двухбайтовое слово размещается в смежных ячейках, причем старший байт занимает ячейку с большим номером. Адресом слова служит адрес младшего байта. Рекомендуется слова размещать по четным адресам.

Хотя все регистры данных, указателей и сегментов 16-разрядные, на шину адреса выдаются и используются при обращениях к ОП 20-разрядные исполнительные (физические) адреса, позволяющие обращаться к ОП емкостью 1 Мбайт. Это становится возможным благодаря механизму сегментации памяти. В адресуемом пространстве выделяются сегменты, содержащие 64 Кбайт. Допускается перекрытие сегментов. Базовые (начальные) 16-разрядные адреса сегментов, хранящиеся в соответствующих сегментных регистрах, трактуются как 20-разрядные с ну-

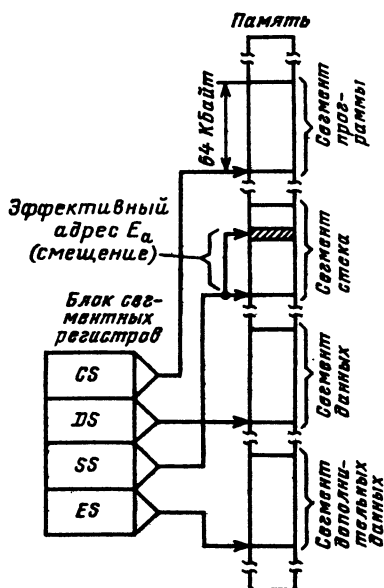


Рис. 10.9. Сегментация адресного пространства памяти

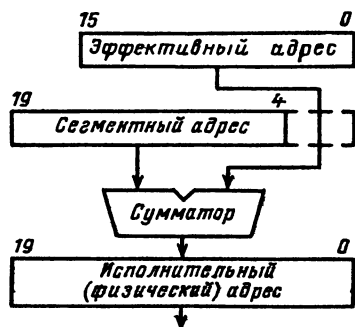


Рис. 10.10. Формирование исполнительного (физического) адреса

лями в четырех младших разрядах. Другими словами, начальные адреса сегментов кратны 16 (рис. 10.9).

Физический адрес операнда или команды формируется, как это показано на рис. 10.10, суммированием начального адреса соответствующего сегмента (содержимого сегментного регистра, сдвинутого на четыре разряда влево и дополненного нулями в четырех младших разрядах) и так называемого эффективного адреса (смещения)  $EА$ , формируемого командой на основе содержащейся в ней информации о способе формирования адреса.

Сегментные регистры позволяют осуществлять динамическое распределение памяти, необходимое для реализации мультипрограммного (многозадачного) режима работы МП (см. гл. 13). Перемещение программ и данных в памяти производится простым изменением сегментных адресов в сегментных регистрах.

Порты ввода-вывода адресуются подобно ячейкам памяти, но без использования сегментных регистров. Доступ к первым 256 портам возможен путем прямой адресации, остальные порты адресуются косвенно по регистру, указываемому командой.

**Обработка прерываний.** Микропроцессор обрабатывает внешние и внутренние (аппаратурные и программные) прерывания. Переход к прерывающей программе производится по присвоенным запросам прерывания номерам векторов прерывания, расположенным в таблице векторов в ОП. При использовании

Рис. 10.11. Форматы команд МП К1810:

*а* — двухоперандная команда; *б* — команда с непосредственно адресуемым операндом; *в* — однооперандная команда; *г*, *д* — однобайтные команды

одной БИС контроллера прерываний возможна обработка до 256 прерываний разных типов: внешние маскируемые (запросы поступают на входы контроллера прерываний); внешние немаскируемые (пропадание напряжения питания, ошибка в ОП и т. п. — запросы поступают на вход незамаскированного прерывания МП); внутренние аппаратные, вызываемые ошибками при делении (переполнение, попытка деления на нуль), а также «пошаговое прерывание», сопровождающее пошаговую работу МП; программное, задаваемое специальными командами, включаемыми в программу (прерывания по переполнению, точкам разрыва и др.).

**Способы адресации.** В командах МП К1810 используются следующие способы адресации: непосредственная, прямая, регистровая прямая и косвенная, относительная (базирование), комбинация базирования и индексации (без смещения и со смещением). При всех способах адресации (кроме непосредственной и регистровой прямой) формируется эффективный адрес *ЕА*, который, как уже указывалось, для получения физического адреса складывается с соответствующим начальным сегментным адресом (рис. 10.10).

**Форматы команд.** На рис. 10.11 представлены примеры форматов команд [25]. Команды имеют длину от 1 до 6 байт. В первом байте команды или в первых двух байтах содержатся код операции и сведения о способе адресации. Команды могут иметь дополнительные (необязательные) байты (показаны на рис. 10.11 штриховыми линиями), задающие: одно (*disp L*)- или двухбайтное (*disp H*, *disp L*) смещение (рис. 10.11, *а*); одно (*data L*)- или двухбайтный (*data H*, *data L*) непосредственный операнд; одно- или двухбайтное смещение, за которым следует одно- или двухбайтный непосредственный операнд

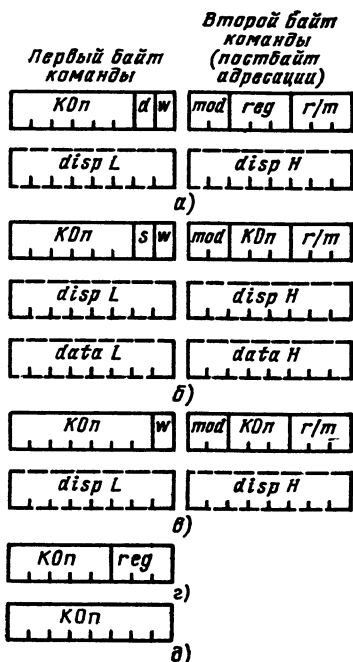


Таблица 10.1

<i>reg (r/m)</i>	<i>w</i> = 0	<i>w</i> = 1	<i>reg (r/m)</i>	<i>w</i> = 0	<i>w</i> = 1
000	<i>AL</i>	<i>AX</i>	100	<i>AH</i>	<i>SP</i>
001	<i>CL</i>	<i>CX</i>	101	<i>CH</i>	<i>BP</i>
010	<i>DL</i>	<i>DX</i>	110	<i>DH</i>	<i>SI</i>
011	<i>B</i>	<i>BX</i>	111	<i>BH</i>	<i>DI</i>

Таблица 10.2

<i>mod</i>	Режим	Пояснения
00	<i>disp</i> = 0	Смещения нет
01	<i>disp</i> = <i>L</i>	8-битное смещение
10	<i>disp</i> = <i>disp H</i> , <i>disp L</i>	16-битное смещение
11	Регистровая адресация в соответствии с табл. 10.1	

Таблица 10.3

<i>r/m</i>	Эффективный адрес (смещение) ( <i>EA</i> )
000	$EA = (BX) + (SI) + disp$
001	$EA = (BX) + (DI) + disp$
010	$EA = (BP) + (SI) + disp$
011	$EA = (BP) + (SI) + disp$
100	$EA = (SI) + disp$
101	$EA = (DI) + disp$
110	$EA = (BP) + disp$
111	$EA = (BX) + disp$

Таблица 10.4

<i>s, w</i>	Формат операнда	Пояснения
00	<i>data L</i>	1 байт данных
01	<i>data H</i> , <i>data L</i>	2 байта данных
11	<i>data L*</i>	1 байт данных с расширением знака до 16 бит

(рис. 10.11, б); двухбайтное смещение и двухбайтный сегментный адрес; двухбайтный эффективный адрес (при прямой адресации).

В двухоперандных командах (рис. 10.11, а) разряд *w* в первом байте команды указывает, выполняется операция над байтами (*w* = 0) или над словами (*w* = 1). Первый операнд находится в памяти или в регистре, а второй — всегда в регистре, номер которого указан в поле *reg* второго байте команды. Присвоен-

ные регистрам номера приведены в табл. 10.1. Адрес операнда можно трактовать как адрес источника или приемника. Результат операции записывается, как правило, по адресу приемника. Разряд  $d$  в первом байте команды определяет, является ли регистр, указываемый в поле *reg*, источником ( $d=0$ ) или приемником ( $d=1$ ).

Способ адресации первого операнда определяется полями *mod* и *r/m* второго бита команды. Если  $mod=11$ , то первый операнд находится в регистре с номером, указанным в поле *r/m* команды. При  $mod \neq 11$  первый операнд находится в памяти, причем поле *mod* задает смещение согласно табл. 10.2, а поле *r/m* — способ формирования эффективного адреса *EA* в соответствии с табл. 10.3. Имеется лишь одно исключение для случая  $mod=00$  и  $r/m=110$ , при котором формируется прямой физический адрес  $EA=disp\ H, disp\ L$ .

Таблицы 10.2 и 10.3 в совокупности показывают, насколько разнообразны используемые в МП К1810 способы адресации.

На рис. 10.11, б представлен формат команды с непосредственно адресуемым операндом, содержащимся в одном или двух дополнительных байтах команды (*data L* или *data H, data L*). Первый операнд адресуется в соответствии с полями *mod* и *r/m*, т. е. как и в предыдущей команде. Результат операции записывается на место первого операнда. Использование непосредственно адресуемого операнда позволило исключить из команды поля  $d$  и *reg*, расширить поле КОп и ввести новое поле *s*. Таблица 10.4 поясняет влияние полей *s* и *w* на формат непосредственно адресуемого операнда.

Формат однооперандной команды (рис. 10.11, в) с учетом сказанного выше в специальных пояснениях не нуждается.

На рис. 10.11, г и д представлены форматы однобайтных команд соответственно с использованием регистровой адресации и с подразумеваемой адресацией одного или двух операндов.

**Особенности структуры микропроцессора Intel 8088.** Структура МП 8088, используемого в ряде моделей персональных компьютеров, представляет собой несколько упрощенный вариант структуры МП 8086 и К1810. Упрощение достигнуто, в первую очередь, за счет уменьшения ширины шины данных до 1 байта (8 линий) вместо 2 байт (16 линий) и некоторых других связанных с более узкой шиной изменений (4-байтный буфер команд вместо 6-байтного, инициирование выборки команды, когда в буфере остался один байт — вместо двух в МП 8086 и К1810 и др.), но при этом сохранены полная программная совместимость этих МП, 20-разрядный адрес, выполнение операций с 16-разрядными операндами, сегментные регистры, средства поддержки мультипрограммного и многопроцессорного режимов.

## 10.4. Новые модели однокристальных 16- и 32-разрядных микропроцессоров (80286, 80386)

Микропроцессорная техника быстро развивается — помимо огромного количественного роста промышленного выпуска микропроцессорных средств наблюдается интенсивное расширение их логических возможностей. Это расширение идет, с одной стороны, путем создания и пополнения наборов микропроцессорных схем, с другой стороны, путем развития архитектуры самих микропроцессоров.

Крупным расширением архитектурных возможностей микропроцессорных средств явилось создание БИС сопроцессоров, т. е. таких БИС, которые, работая параллельно и совместно с БИС центрального микропроцессора, разгружают его от выполнения ряда функций, которые он может выполнять лишь с помощью подпрограмм. Эти функции выполняются сопроцессорами с высокой скоростью параллельно с работой центрального микропроцессора. Такая комбинация микропроцессорных БИС существенно повышает производительность и точность вычислений микропроцессорной системы.

С однокристальным 16-разрядным микропроцессором могут работать:

*арифметический 80-разрядный сопроцессор*, на два порядка повышающий скорость выполнения высокоточных операций с плавающей точкой, а также выполняющий тригонометрические, экспоненциальные и логарифмические команды;

*сoproцессор ввода-вывода*, осуществляющий функции контроллера с двумя каналами прямого доступа к памяти, что позволяет выполнять операции ввода-вывода параллельно с выполнением вычислений центральным микропроцессором;

*сoproцессоры ядра операционной системы*, содержащие на кристалле в ПЗУ ядро ОС, а также ряд таймеров, блоки управления и прерывания и позволяющие достигнуть значительного ускорения работы операционной системы.

Некоторые из схем, входящих в набор микропроцессорных БИС, по своей сложности не уступают, а в ряде случаев могут превосходить БИС самих микропроцессоров.

Новые микропроцессорные средства с более развитой архитектурой и более высокой производительностью создаются на основе совершенствования технологии БИС и СБИС, в том числе путем уменьшения размеров схемных компонентов, использования технологии комплементарных МОП-транзисторов, позволяющих снизить на порядок потребляемую мощность, расширить диапазон рабочих температур аппаратуры.

В качестве примеров МП следующих (по сравнению с МП

K1810) поколений рассмотрим микропроцессоры 80286 и 80386 фирмы Intel, широко используемые в современных персональных компьютерах. В этих МП реализованы многие логические свойства, которые еще недавно считались принадлежностью лишь крупных ЭВМ (защита памяти, кэш-память, сегментированная виртуальная память и др.). Для лучшего понимания содержания данного параграфа полезно ознакомиться с материалами гл. 13 и 14.

Однокристалльный 16-разрядный МП 80286 является существенным развитием МП 8086. Кристалл МП 80286 содержит 130 тыс. транзисторов, заключен в квадратный корпус с 68 контактами, расположенными по четырем сторонам корпуса. Используется один уровень питающего напряжения +5 В.

Микропроцессор 80286 содержит схемы управления и защиты памяти. Тактовая частота 8 или 10 МГц. Производительность этого МП в 2—3 раза выше, чем у МП 8086.

В МП реализуется конвейерная обработка команд и данных, многозадачный (мультипрограммный) и многопользовательский режимы работы. Емкость виртуальной памяти достигает  $2^{30}$ , или 1 Гбайт, а физической  $2^{24}$ , или 16 Мбайт.

При подключении к МП 80286 80-разрядного арифметического сопроцессора 80287 достигается значительное увеличение точности и скорости вычислений с числами с плавающей точкой, определения значений элементарных функций.

Программистская модель МП 80286 представлена на рис. 10.12 [35]. Она содержит наряду с регистрами, присутствующими в МП 8086, несколько дополнительных регистров, поддерживающих мультипрограммные и многопользовательские режимы.

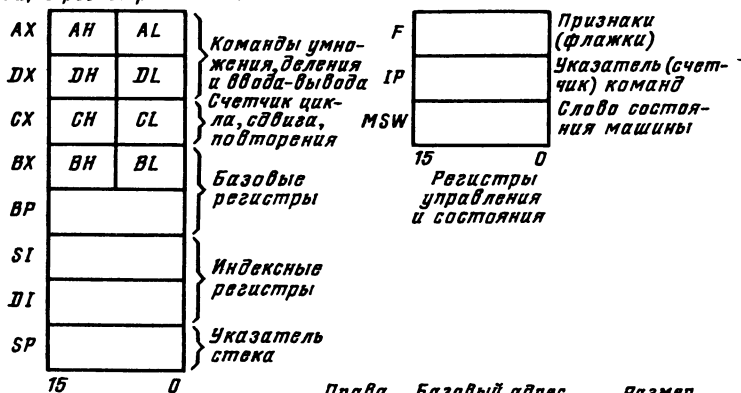
Расширение средств отображения состояния МП достигнуто добавлением к обычному для структуры МП регистра флажков *F* регистра слова состояния машины — *MSW*. В частности, разряд разрешения запрета в этом регистре определяет альтернативные режимы работы МП: режим реального адреса или режим защищенного виртуального адреса.

Введены сегментные дескрипторы (см. гл. 9), описывающие размеры сегментов, права доступа, их базовые адреса. Имеется дескрипторная таблица. Регистры дескрипторной таблицы — сегментные селекторы (*LDTR*, *GDTR*, *IDTR*) — указывают адреса дескрипторных таблиц.

Регистр задачи *TR* обслуживает переключение задач. Этот регистр содержит адрес системного дескриптора сегмента, задающего область запоминания состояния машины (задачи) в момент перехода задачи из активного состояния в состояние ожидания. Переход к обработке новой задачи осуществляется



# Общие регистры данных



Сегментные селекторы

CS
DS
SS
ES

15 0

Сегментные регистры (загружаются программой)

Сегментные селекторы  
15 0

TR
LDT
GDTR
IDTR

Регистры, загружаемые операционной системой и процессором

Права доступа	Базовый адрес сегмента	Размер сегмента
7 0 23	0 15	0
47 40 39	16 15	0

47 40 39 15 15 0  
Регистры кэш-памяти сегментных дескрипторов (процессор загружает эту кэш-память)

Права доступа	Базовый адрес сегмента	Размер сегмента

47 40 39 16 15 0  
Процессор загружает эту память

Регистры задачи и регистры дескрипторной таблицы

Рис. 10.12. Программистская модель однокристального 16-разрядного микропроцессора Intel 80286

передачей в регистр задачи адреса системного дескриптора сегмента, определяющего область запоминания состояния новой задачи, откуда это состояние загружается в соответствующие регистры МП.

В режиме реального адреса дополнительные регистры не используются и МП 80286 работает как МП 8086.

Микропроцессор 80286 выполняет все команды МП 8086, а также ряд новых команд: команды загрузки и запоминания регистров дескрипторных таблиц, регистра слова состояния машины, регистра задачи; команды, расширяющие состав операций со стеком (включение в стек и извлечение из стека содержимого группы регистров и др.); команды входа и выхода из процедуры, облегчающие реализацию языков программирования высокого уровня.

*32-разрядный однокристалльный микропроцессор Intel 80386.*  
32-разрядный однокристалльный МП 80386 — новый крупный шаг в развитии технологии и архитектуры микропроцессорных средств [8, 88]. Он изготавливается по КМОП-технологии с проектной нормой на ширину проводников 1,5 мкм, позволившей на кристалле площадью примерно 100 мм<sup>2</sup> разместить около 275 000 транзисторов. Кристалл находится в керамическом корпусе со 132 выводами. Тактовая частота 12 или 16 МГц, рассеиваемая мощность не превышает 2 Вт.

К основным особенностям архитектуры МП 80386 следует отнести:

- наличие средств, обеспечивающих реализацию мультипрограммного (многозадачного) и многопользовательского режимов работы МП и режима «системы виртуальных машин» (см. гл. 13), при котором пользовательские программы могут выполняться параллельно во времени под управлением разных операционных систем;

- непосредственный доступ к физическому адресу пространству в 4 Гбайт и виртуальной памяти емкостью 64 Тбайт (примерно 70 триллионов байт);

- сегментно-страничная организация памяти;

- высокая производительность — в 2—3 раза превосходит производительность МП 80286 и достигается за счет большей тактовой частоты, более быстрого доступа к памяти благодаря использованию размещенных на кристалле МП кэш (скрытой)-памяти и блока управления и защиты памяти (в том числе блока быстрого преобразования адресов);

- система команд МП является расширением системы команд МП 8086 (аналога МП 1810), обеспечивается программная совместимость с МП 8086 и 80286 (на уровне двоичных кодов программ);

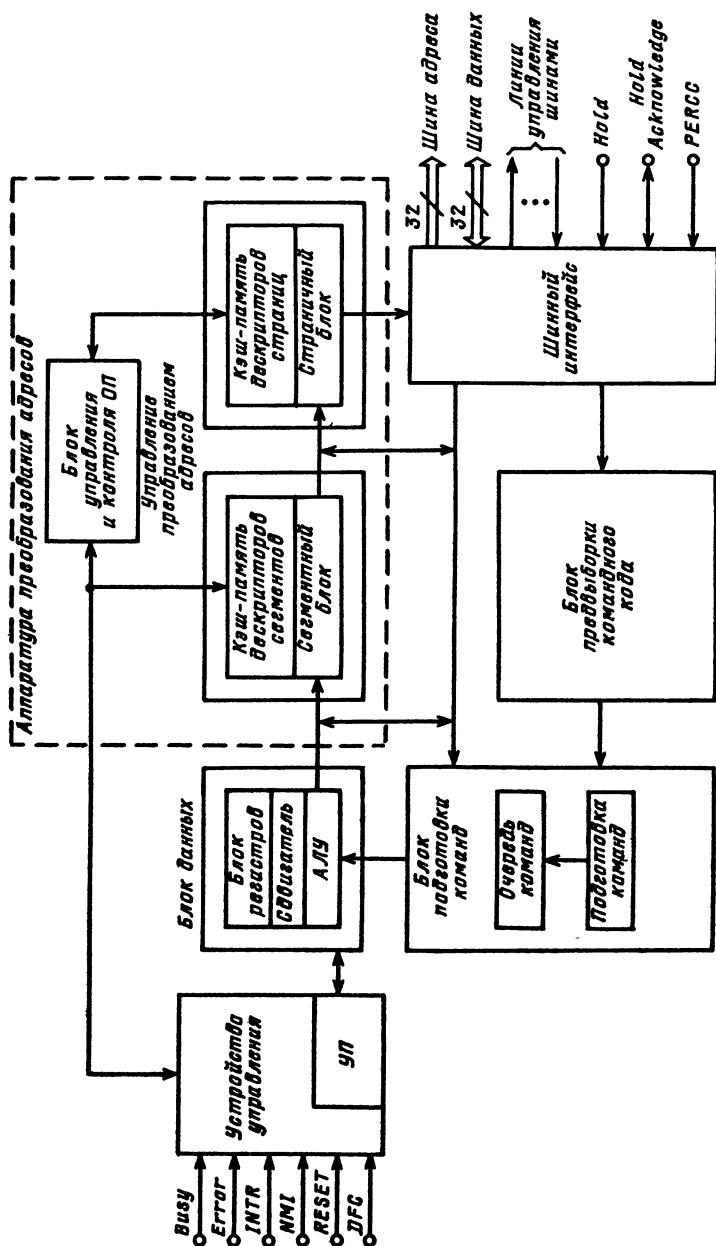


Рис. 10.13. Структура однокристалльного 32-разрядного микропроцессора Intel 80386

обработка данных различных типов: целые числа, числа с плавающей точкой, десятичные числа, байты, строки символов, цепочки бит до 4 Гбит;

использование в командах 32-, 16-, 8-разрядных операндов; наличие встроенных средств самотестирования, запускаемых сигналом сброса и проверяющих примерно 75 % всех транзисторов, расположенных на кристалле.

Упрощенная структура МП 80386 приведена на рис. 10.13. Блок данных содержит восемь 32-разрядных общих регистров. В целях создания условий для выполнения операций с 16- и 8-разрядными словами (в том числе для совместимости с МП 8086 и 80286) в каждом общем регистре адресуемо младшее полуслово, а в четырех 16-разрядных регистрах адресуемы в отдельности старший и младшие байты.

Для повышения быстродействия МП в блок данных введены 64-разрядный сдвиговый регистр («сдвигатель») и аппаратурные средства ускоренного выполнения операций умножения и деления.

Вектор состояния процессора образуют содержимые 32-разрядных счетчика команд (смещение адреса команды относительно базового адреса) и регистра признаков (флажков). В регистре признаков формируются три группы признаков: признаки результата (знака результата, нуля, переноса, переполнения и др.); признаки управления (направление и др.); системные признаки (разрешение прерывания, режим виртуальной памяти, порог прерывания и др.). Отметим, что около половины разрядов регистра флажков не используется.

Микропроцессор через шинный интерфейс имеет доступ к внешним 32-разрядной шине адреса, 32-разрядной двунаправленной шине данных, линиям управления шинами, линиями: захват (Hold), подтверждение захвата (Hold Acknowledge) и запроса сопроцессора (Processor Extension Request Coprocessor Control — PERCC).

Устройство управления (УУ), содержащее управляющую память (УП) микропрограмм, с учетом внешних сигналов (занят — Busy, ошибка — Error, прерывание — INTR, NMI, сброс — Reset, двойная тактовая частота — Double Frequency Clock — DFC) вырабатывает управляющие сигналы, инициирующие соответствующие микрооперации.

В МП выполняется конвейерная обработка команд на восьми позициях, образованных восемью его основными блоками, представленными на рис. 10.13.

Используемая пользователем виртуальная память может быть разделена на несколько сегментов, каждый размером до 4 Гбайт. Сегменты состоят из страниц, содержащих 4 Кбайта.

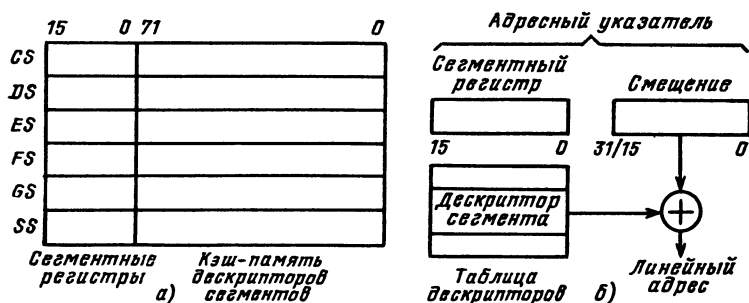


Рис. 10.14. Организация сегментирования

При сегментно-страничной организации виртуальной памяти используются расположенные на кристалле МП сегментные регистры и быстродействующие (скрытые от пользователя) кэш-памяти, хранящие дескрипторы (описатели) сегментов и страниц. Так, дескриптор сегмента определяет базовый адрес сегмента, адресные границы, условия защиты и другие данные (рис. 10.14, а).

Имеются следующие регистры: *CS* — сегмента командных кодов; *DS* — сегмента данных; *SS* — сегмента стека; *ES*, *FS* и *GS* — дополнительных сегментов данных.

Сегментные регистры хранят «селекторы», адресующие соответствующие дескрипторы в кэш-памяти таблицы дескрипторов сегментов. Сами сегментные регистры в командах явно не адресуются (подразумеваемая адресация).

Формирование «линейного адреса» в сегментированной памяти поясняется на рис. 10.14, б. Адресный указатель, образованный смещением в команде и содержащимся в сегментном регистре селектором, определяющим соответствующий дескриптор сегмента, преобразуется в 32-разрядный линейный адрес. На этом заканчивается первый этап преобразования адреса.

Общий случай преобразования линейного адреса в фактический (адрес в физической памяти) показан на рис. 10.15, а.

Из справочника, начальный адрес которого хранит регистр управления страницами, выбирается адрес начала страничной таблицы. Затем из страничной таблицы выбирается начальный адрес страницы, который суммируется со смещением, образуя физический адрес. Может оказаться, что нужной страницы нет в ОП («страничный сбой»). Тогда по прерыванию нужная страница передается из внешней памяти в ОП.

Сокращение времени, расходуемого на преобразование адресов, достигается при помощи блока ускоренного преобразования, хранящего сведения о физических адресах для нескольких

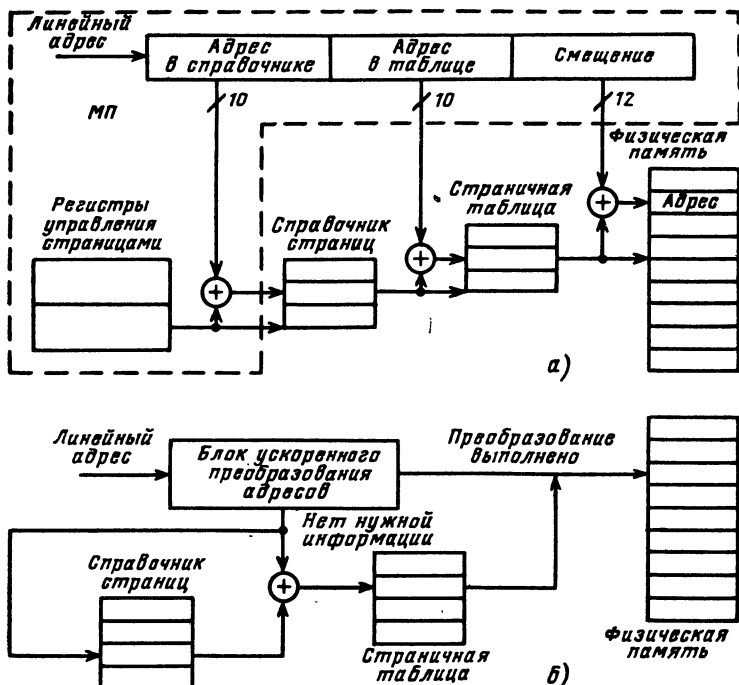


Рис. 10.15. Преобразование линейного адреса:

а — с использованием справочника и таблиц; б — с использованием блока ускоренного преобразования адресов

недавно использовавшихся страниц (рис. 10.15, б). В этом случае преобразование адресов для этих страниц происходит без обращений к справочнику и страничной таблице. Если нужной информации нет в блоке, преобразование происходит в соответствии с рис. 10.15, а. Общие вопросы организации виртуальной памяти рассмотрены в гл. 14.

Микропроцессор 80386 в состоянии одновременно выполнять программы, предназначенные для МП 8086, 80286 и 80386. В МП 80386 возможны два режима работы: реальный режим и защищенный режим виртуальной памяти. В реальном режиме МП эмулирует с повышенной скоростью МП 8086/8088, работая в однопрограммном режиме с адресным пространством, ограниченным 1 Мбайт.

В защищенном режиме МП может использовать все свое адресное пространство и реализовывать «систему виртуальных 8086-машин» с распределением памяти согласно рис. 10.16.

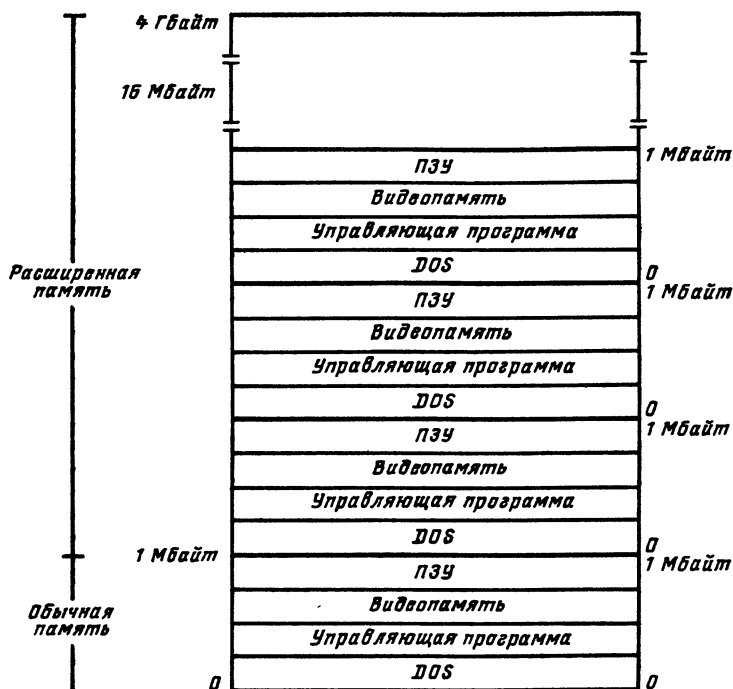


Рис. 10.16. Режим виртуальных процессоров 8086

Каждый пользователь виртуального МП 8086 получает для своей задачи 1 Мбайт в расширенной памяти, при этом задачи защищены друг от друга средствами защиты памяти, реализуемыми с помощью аппарата дескрипторов сегментов и страниц. Однако возникают некоторые затруднения при попытках одновременного использования несколькими программами МП 8086 периферийных устройств, в первую очередь экрана дисплея. Вопрос может решаться путем применения специальной программы «виртуальный монитор», перехватывающей обращения к операциям ввода-вывода программ виртуальных МП 8086 и организующей «виртуальный экран».

Используя микропроцессоры с архитектурой и характеристиками, подобными МП 80386, можно создавать сравнительно недорогие супермикроЭВМ для применений, требующих больших объемов вычислений, операций с крупными базами данных, работы с большими электронными таблицами, обработки графической информации, в том числе для использования в инженерных АРМ при автоматизации проектирования, а также для

реализации экспертных систем, при построении устройств, управляющих системой информационных файлов (файловых серверов) локальных сетей (см. гл. 16).

## 10.5. Персональные компьютеры

В последние годы интенсивные взаимосвязанные процессы развития технологии интегральных микросхем с большой и сверхбольшой степенями интеграции (в первую очередь, микропроцессорных средств), архитектуры ЭВМ и принципов построения программного обеспечения ознаменовались революционным скачком в вычислительной технике — созданием персональных компьютеров.

*Персональный компьютер* (ПК) — малогабаритный, с малой потребляемой мощностью, дешевый, высокопроизводительный и высоконадежный комплекс вычислительной аппаратуры и программных средств, ориентированный на индивидуальное использование массовым пользователем, не имеющим специальной подготовки в области вычислительной техники и программирования.

Важнейшее свойство ПК — простота их использования — достигается эффективными средствами общения пользователя с компьютером (см. гл. 5), и в первую очередь, «дружелюбным» характером программного обеспечения, позволяющим широко применять диалоговый режим работы с активным участием в нем самого компьютера путем внесения в процесс общения в той или иной степени игрового компонента, представления пользователю для выбора различных «меню», подсказок при его ошибках, реализации различных обучающих программ.

На рис. 10.17 представлена структурная схема персональных профессиональных компьютеров ЕС-1840 и его расширенного варианта ЕС-1841 [44]. Эти ПК построены на основе рассмотренного в § 10.3 16-разрядного микропроцессора K1810BM86.

Компьютер состоит из нескольких электронных модулей, подключаемых к системной шине, и периферийных устройств (назначение модулей и типы периферийных устройств обозначены на рисунке). Основной блок и блок расширения могут содержать до семи модулей каждый. Набор модулей может варьироваться, в том числе за счет дополнительных системных модулей и специальных модулей профессиональной ориентации, образуя различные конфигурации персонального компьютера. Блоки (основной и расширения) снабжаются источниками питания и динамиками. Динамик управляется сигналами от таймера или от программируемого интерфейса периферийного оборудования.



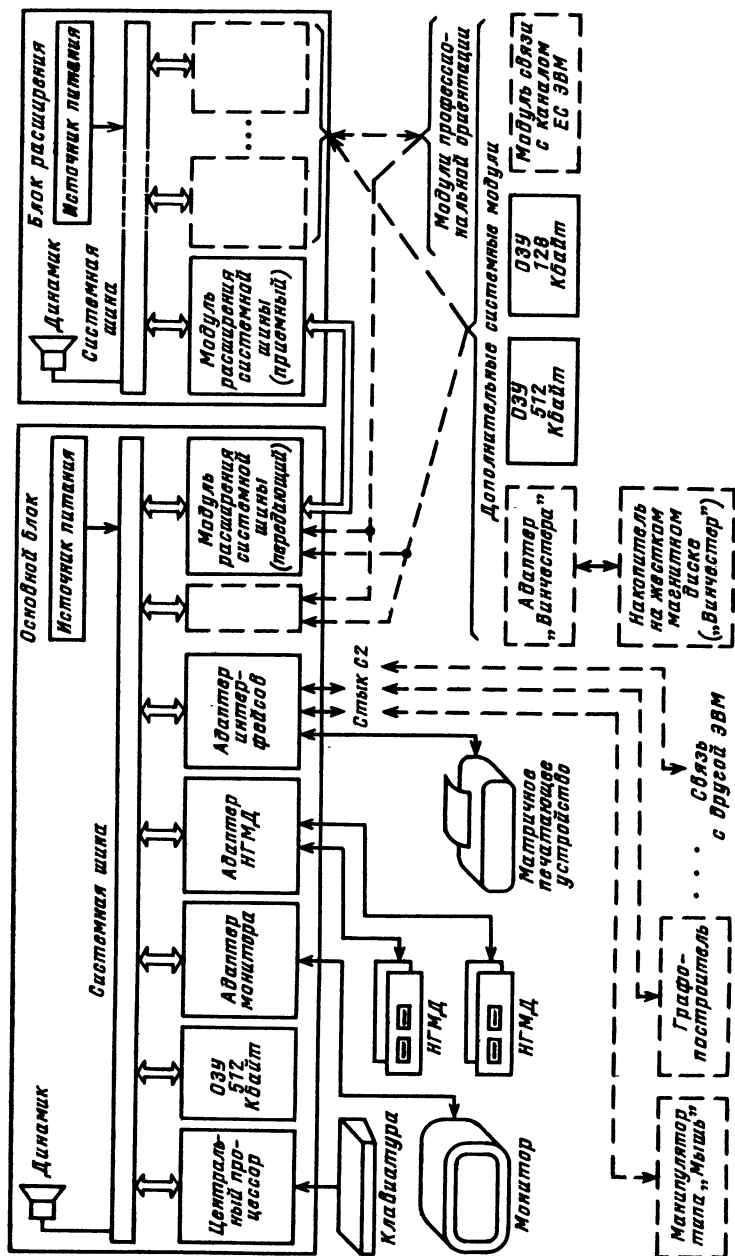


Рис. 10.17. Структура персональных компьютеров ЕС-1840 и ЕС-1841

Системный модуль (центральный процессор) помимо микропроцессора K1810BM86 содержит постоянную память, таймер, контроллеры прерываний, прямого доступа к памяти, схемы управления системной шиной и генерации синхросигналов. В постоянной памяти хранятся программа инициализации (установка в начальное состояние аппаратуры ПК, программная настройка контроллера прерываний и прямого доступа к памяти, адаптеров, таймера), программа тестирования процессора, памяти, адаптеров, программа управления вводом-выводом.

Предусмотрена возможность работы системного модуля с арифметическим сопроцессором K1810BM87.

Базовая конфигурация ПК ЕС-1840 содержит основной блок с системным модулем, модулем ОЗУ 512 Кбайт, модулями адаптеров монитора, НГМД и интерфейсов, а также клавиатуру, монохромный дисплей, матричное печатающее устройство, два ЗУ на гибких магнитных дисках.

Базовая конфигурация ПК ЕС-1841 помимо того, что имеет базовая конфигурация ПК ЕС-1840, включает в себя модули адаптера ЗУД типа «Винчестер», графического дисплея (вместо адаптера монитора), а также монохромный или цветной графический дисплей, манипулятор «мышь» для управления движущимися графическими изображениями.

Особо следует отметить наличие среди дополнительных системных модулей адаптера кольцевой локальной сети (см. гл. 16), позволяющего подключать персональные компьютеры к сети и при наличии соответствующего сетевого программного обеспечения организовывать их взаимодействие (например, режим «электронной почты»).

В табл. 10.5 приведены основные характеристики персональных компьютеров Единой системы ЭВМ [44].

Персональные компьютеры, выпускаемые промышленностью в сотнях тысяч и миллионах экземпляров, вносят коренные изменения в формы использования вычислительных средств, в огромной степени расширяют масштабы их применения. Они находят широкое применение как для поддержки различных видов профессиональной деятельности (инженерной, административной, литературной, врачебной, производственной и др.), так и в быту, например для обучения в различных областях знаний и для досуга (электронные игры).

Персональный компьютер позволяет эффективно выполнять научно-технические и планово-экономические расчеты, организовывать базы данных, подготавливать и редактировать документы и любые другие тексты, вести делопроизводство, обрабатывать таблицы, результаты экспериментов, решать задачи автоматизированного проектирования, представлять результаты

**Т а б л и ц а 10.5. Основные характеристики персональных ПК ЕС-1840 и ЕС-1841**

Характеристики	ЕС-1840	ЕС-1841
Микропроцессор	16-разрядный МП К1810ВМ86, тактовая частота 4 МГц, быстродействие 1 млн. операций/с (типа регистр-регистр)	
Емкость основной памяти, Кбайт	512	512—1536
ЗУ на гибких дисках:		
количество ЗУ	2	2
диаметр дискеты, мм	133	133
полезная емкость дискеты, Кбайт	320	320
ЗУ на жестких дисках («Винчестер»):		
диаметр диска, мм	—	133
полезная емкость диска, Мбайт	—	10
Дисплей типа I:	Алфавитно-цифровой мо-нохромный	Цветной гра-фический
диагональ экрана, см	31	32
разрешающая способность	80 знаков × × 25 строк	640 × 200 точек
количество цветов	—	16
Дисплей типа II:	Монохромный, графический	
диагональ экрана, см	31	
разрешающая способность	640 × 200 точек	
количество градаций яркости	16	
Печатающее устройство:	Матричный	
способ печати	До 132	
количество знаков в строке	До 160 знаков/с	
скорость печати	От сети переменного тока 220 В (от +10 до —15 %)	
Питание		
Потребляемая мощность, кВт, не более	0,2	0,22
Масса ПК, кг, не более	40	42
Климатические условия эксплуатации:		
температура окружающей среды, °С	От 10 до 35	
относительная влажность воздуха, %, при 25 °С	40—80	
атмосферное давление, КПа	84—107	

расчетов в цифровой и графической форме, обрабатывать графические изображения.

На основе ПК создаются *автоматизированные рабочие места* (АРМ) разных профессий (конструкторов, инженеров-расчетчиков, технологов, работников административного аппарата и др.).

Выполнение многих из указанных функций поддерживается многочисленными эффективными универсальными функциональными пакетами программ. Примером является ориентированный на АРМ работников административного аппарата интегрирован-

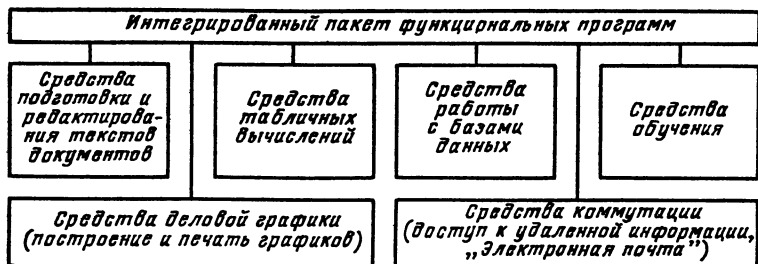


Рис. 10.18. Структура интегрированного пакета функциональных программ персонального компьютера АРМ сотрудника учреждения

ный пакет, охватывающий функции редактора текста, формирования и управления базой данных, выполнения операций над электронными таблицами. Структура пакета приведена на рис. 10.18.

## 10.6. Новые персональные вычислительные средства — персональные системы

В мировой вычислительной технике в настоящее время наблюдается интенсивное развитие персональных вычислительных средств — повышается производительность персональных вычислительных компьютеров за счет новых архитектурных решений и увеличения тактовой частоты микропроцессора до 16—30 МГц. Увеличилась до 2—16 Мбайт емкость основной памяти, до 70—100 Мбайт емкость ЗУ на жестких магнитных («винчестерских») дисках, уменьшаются размеры ПК, появились переносимые модели ПК («ПК в портфеле»), широкое развитие получили разработка и производство плат профессиональной ориентации ПК, в комплект ПК включаются средства его подключения к локальным сетям. Создают новые операционные системы для персональных вычислительных средств, поддерживающие многозадачные и многопользовательские режимы и сетевое взаимодействие ПК, различные функциональные пакеты программ. Большое внимание уделяется улучшению характеристик графических аппаратных и программных средств, имеющих первостепенное значение для развития различных АРМ.

Основу для перспективных персональных вычислительных средств создают новые высокопроизводительные микропроцессоры, например, такие, как 16-разрядный 80286 и 32-разрядный 80386, используемые, в частности, в серии «Персональные системы» PS/2 фирмы IBM, которая рассматривается как новый этап в развитии персональных компьютеров.

**Т а б л и ц а 10.6. Характеристики персональных систем IBM PS**

Модель	Микро-процессор (тактовая частота)	Максимальная производительность	Емкость памяти, Мбайт		Гибкий диск		Емкость ЗУ на жестких дисках, Мбайт	Операционная система
			Номинальная	Расширенная	Размер, мм	Емкость, Мбайт		
PS/2-30	8086 (8 МГц)	До 2,5 раза (PC XT)	0,640	—	89	0,720	20	PS DOS 3.3
PS/2-50	80286 (10 МГц)	До 2 раз (PC AT)	1	7	89	1,44	20	PS DOS 3.3 OS/2
PS/2-60	80286 (10 МГц)	До 2 раз (PC AT)	1	15	89	1,44	44; 70; 115	PS DOS 3.3 OS/2
PS/2-80	80386 (16 или 20 МГц)	До 3,5 раза (PC AT)	До 2	16	89	1,44	44; 70; 115	PS DOS 3.3 OS/2
PC AT	80286 (8 МГц)	—	0,256	0,512	113	1,2	20	PS DOS 3,0; 3.1; 3.2

В табл. 10.6 приведены характеристики моделей персональных систем PS/2 [41].

Важным достоинством моделей PS/2 являются развитые графические средства, сочетающие встроенную графическую аппаратуру и графический интерфейс пользователя на основе пакета программ Windows, что, по-видимому, станет новым графическим стандартом для персональных компьютеров. Эти графические средства VGA (Video Graphics Array — видеографическая матрица) имеют разрешающую способность 640××480 элементов отображения, причем все машины, кроме модели 30, снабжаются цветными дисплеями.

В PS/2 (кроме модели 30) применена новая 32/16-битная шина расширения — «Микроканал».

Наиболее полно вычислительные и графические возможности моделей персональных систем PS/2 раскрываются при работе в среде многозадачной операционной системы OS/2, позволяющей МП 80286 и 80386 адресовать память максимальной емкости (16 Мбайт).

### **Контрольные вопросы**

1. Каковы особенности трехшинной структуры микропроцессора и организации связи между блоками?

2. Сравните программистские модели МП К580 и процессоров малых ЭВМ (см. гл. 9). Что в них общего и каковы различия? Какова роль в повышении эффективности МП его архитектурных элементов: средств организации стековой памяти, схемы инкрементатора и декрементатора, регистра косвенного адреса, механизма тандемных передач?

3. Какие особенности построения регистра-аккумулятора позволяют использовать его одновременно в качестве регистра операнда и регистра результата?

4. Что общего и в чем различие в организации векторных прерываний в МП К580 и в малых ЭВМ (см. гл. 9)?

5. Поясните механизм сегментации памяти в МП К1810 и его роль в расширении адресного пространства микропроцессора.

6. Какие архитектурные особенности МП Intel 80386 поддерживают реализацию многопрограммного и многопользовательского режимов работы? При подготовке ответа следует воспользоваться материалами гл. 13 и 14.

7. Каковы назначение и принципы функционирования периферийных устройств, входящих в состав персонального компьютера (см. рис. 10.17)? При подготовке ответа следует использовать материалы § 5.5.

## Глава 11

# ПРИНЦИПЫ ОРГАНИЗАЦИИ СИСТЕМ ВВОДА-ВЫВОДА. ИНТЕРФЕЙСЫ ЭВМ И МИКРОПРОЦЕССОРОВ

## 11.1. Проблемы организации систем ввода-вывода

Вычислительная машина содержит помимо процессора (процессоров) и основной памяти, образующих ее ядро, многочисленные и разнообразные по выполняемым функциям и принципам действия *периферийные устройства* (ПУ), предназначенные для хранения больших объемов информации (*внешние запоминающие устройства*) и для ввода в ЭВМ и вывода из нее информации, в том числе для ее регистрации и отображения (*устройства ввода-вывода*).

Передача информации с периферийного устройства в ядро ЭВМ (память и процессор) называется *операцией ввода*, а передача из ядра ЭВМ в периферийное устройство — *операцией вывода*.

Производительность и эффективность использования ЭВМ определяются не только возможностями ее процессора и характеристиками основной памяти, но в очень большой степени составом ее ПУ, их техническими данными и способом организации их совместной работы с ядром (процессором и основной памятью) ЭВМ.

Связь устройств ЭВМ друг с другом осуществляется с помощью сопряжений, которые в вычислительной технике называются интерфейсами.

*Интерфейс* представляет собой совокупность линий и шин, сигналов, электронных схем и алгоритмов (протоколов), предназначенную для осуществления обмена информацией между устройствами. От характеристик интерфейсов во многом зависят производительность и надежность вычислительной машины.

При разработке систем ввода-вывода должны быть решены следующие проблемы:

должна быть обеспечена возможность реализации машин с переменным составом оборудования (*машин с переменной конфигурацией*), в первую очередь, с различным набором периферийных устройств, с тем чтобы пользователь мог выбирать состав оборудования (конфигурацию) машины в соответствии с ее назначением, легко дополнять машину новыми устройствами;

для эффективного и высокопроизводительного использования оборудования ЭВМ должны реализовываться параллельная во времени работа процессора над программой и выполнение периферийными устройствами процедур ввода-вывода;

необходимо упростить для пользователя и стандартизировать программирование операций ввода-вывода, обеспечить независимость программирования ввода-вывода от особенностей того или иного периферийного устройства;

необходимо обеспечить автоматическое распознавание и реакцию ядра ЭВМ на многообразие ситуаций, возникающих в ПУ (готовность устройства, отсутствие носителя, различные нарушения нормальной работы и др.).

Особенно актуально решение этих проблем для машин средней и большой производительностей, содержащих большое число разнообразных периферийных устройств.

Отметим основные пути решения указанных проблем.

*Модульность.* Средства современной вычислительной техники проектируются на основе модульного (или агрегатного) принципа, который заключается в том, что отдельные устройства выполняются в виде конструктивно законченных модулей (агрегатов), которые могут сравнительно просто в нужных количествах и номенклатуре объединяться, образуя вычислительную машину.

Присоединение нового устройства не должно вызывать в существующей части машины никаких других изменений, кроме изменения кабельных соединений и некоторых корректировок программ.

*Унифицированные* (не зависящие от типа ПУ) *форматы данных*, которыми ПУ обмениваются с ядром ЭВМ, в том числе

унифицированный формат сообщения, которое ПУ посылает в ядро о своем состоянии. Преобразование унифицированных форматов данных в индивидуальные, приспособленные для отдельных ПУ, производится в самих ПУ, точнее, в блоках управления ПУ (УПУ).

*Унифицированный интерфейс*, т. е. унифицированный по составу и назначению набор линий и шин, унифицированные схемы подключения, сигналы и алгоритмы (протоколы) управления обменом информацией между ПУ и ядром ЭВМ.

*Унифицированные* (не зависящие от типа ПУ) *формат и набор команд* процессора для операций ввода-вывода. Операция ввода-вывода с любым ПУ представляет для процессора просто операцию передачи данных независимо от особенностей принципа действия данного ПУ, типа его носителя и т. п.

Унификация распространяется на семейство (ряд, систему) моделей ЭВМ.

Для обеспечения параллельной во времени работы ПУ с выполнением программы обработки данных процессором схемы управления вводом-выводом отделяют от процессора и придают им достаточную степень автономности.

Многие функции управления операциями ввода-вывода, как, например, управление прямым доступом к памяти (см. § 11.2), являются общими, они не зависят от типа ПУ. Другие являются специфичными для данного типа устройств.

Выполнение общих функций возлагают на общие для групп периферийного оборудования унифицированные устройства — контроллеры прямого доступа к памяти, процессоры (каналы) ввода-вывода, а специфических — на специализированные для данного типа ПУ электронные блоки управления (УПУ), часто называемые адаптерами.

## 11.2. Прямой доступ к памяти

В системах ввода-вывода ЭВМ используются два основных способа организации передачи данных между памятью и периферийными устройствами: программно-управляемая передача и прямой доступ к памяти (ПДП).

*Программно-управляемая передача данных* (рис. 11.1, а) осуществляется при непосредственном участии и под управлением процессора, который при этом выполняет специальную подпрограмму ввода-вывода. Данные между памятью и периферийным устройством пересылаются через процессор. Операция ввода-вывода инициируется текущей командой программы или запросом прерывания от периферийного устройства.



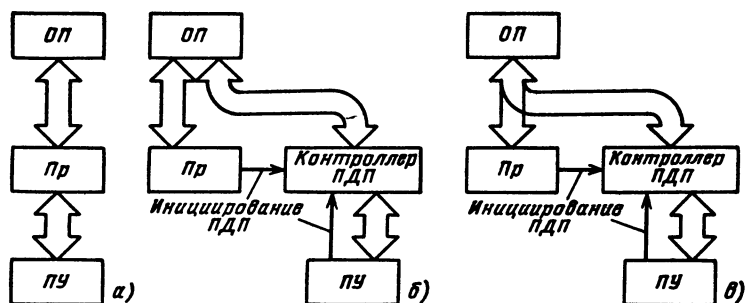


Рис. 11.1. Способы организации передачи данных между памятью и периферийными устройствами:

а — программно-управляемая передача; б — прямой доступ к памяти при наличии отдельной шины в памяти для ПДП; в — прямой доступ к памяти при использовании процессором и ПДП одной шины для связи с памятью

При программно-управляемой передаче данных процессор на все время этой операции отвлекается от выполнения основной программы решения задачи. Операция пересылки данных логически слишком проста, чтобы эффективно загружать логически сложную быстродействующую аппаратуру процессора. В результате при использовании программно-управляемой передачи данных снижается производительность вычислительной машины.

Вместе с тем при пересылке блока данных процессору приходится для каждой единицы передаваемых данных (байт, слово) выполнять довольно много команд, чтобы обеспечить буферизацию данных, преобразование форматов, подсчет количества переданных данных, формирование адресов в памяти и т. п. В результате скорость передачи данных при пересылке блока данных даже через высокопроизводительный процессор не превышает 0,02—0,05 Мбайт/с, что недостаточно для работы с высокоскоростными периферийными устройствами (например, с ЗУ на дисках и барабанах, с аналого-цифровыми преобразователями и т. п.), и может оказаться вообще неприемлемой для систем управления, работающих в реальном времени. Между тем потенциально возможная максимальная скорость обмена данными при вводе-выводе определяется пропускной способностью памяти, которая, например, при цикле памяти 0,5 мкс составляет  $2b$  Мбайт/с, где  $b$  — ширина выборки, измеряемая числом байт, одновременно записываемых (считываемых) в памяти.

Для быстрого ввода-вывода блоков данных и разгрузки процессора от управления операциями ввода-вывода используют прямой доступ к памяти.

*Прямый доступ к памяти* называется способ обмена дан-

ными, обеспечивающий автономно от процессора установление связи и передачу данных между ОП и ПУ (рис. 11.1, б и в).

Прямой доступ к памяти освобождает процессор или микропроцессор от управления операциями ввода-вывода, позволяет осуществлять параллельно во времени выполнение процессором (микропроцессором) программы с обменом данными между периферийным устройством и ОП, производить этот обмен со скоростью, ограничиваемой только пропускной способностью ОП или ПУ. Таким образом, ПДП, разгружая процессор (микропроцессор) от обслуживания операций ввода-вывода, способствует возрастанию общей производительности ЭВМ или микроЭВМ. Повышение предельной скорости ввода-вывода информации делает машину более приспособленной для работы в системах реального времени. Прямым доступом к памяти управляет контроллер ПДП, который выполняет следующие функции:

управление иницилируемой процессором или ПУ передачей данных между ОП и ПУ;

задание размера блока данных, который подлежит передаче, и области памяти, используемой при передаче;

формирование адресов ячеек ОП, участвующих в передаче;

подсчет числа единиц данных (байт, слов), передаваемых от ПУ в ОП или обратно, и определение момента завершения заданной операции ввода-вывода.

Указанные функции реализуются контроллером ПДП обычно с помощью одного или нескольких буферных регистров  $RzB$ , регистра — счетчика текущего адреса данных  $RzTAD$  и счетчика текущих данных  $TсчД$  (рис. 11.2).

При иницировании операции ввода-вывода в  $TсчД$  заносится размер подлежащего передаче блока (число байт или слов), а в  $RzTAD$  — начальный адрес области памяти, используемой при передаче. При передаче каждого байта содержимое  $RzTAD$  увеличивается на 1, при этом формируется адрес очередной ячейки ОП, участвующей в передаче. Одновременно уменьшается на 1 содержимое  $TсчД$ . Обнуление  $TсчД$  указывает на завершение передачи. Контроллер ПДП обычно имеет более

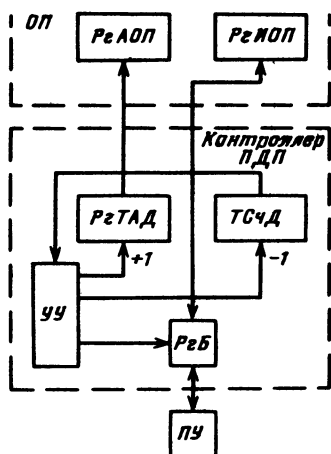


Рис. 11.2. Структурная схема контроллера прямого доступа к памяти

высокий приоритет в занятии цикла памяти по сравнению с процессором. Управление памятью переходит к контроллеру ПДП, как только завершится цикл ее работы, выполняемый для текущей команды процессора.

Прямой доступ к памяти обеспечивает высокую скорость обмена данными за счет того, что управление обменом производится не программным путем, а аппаратными средствами.

В крупных и средних ЭВМ ПДП является основным способом осуществления операций ввода-вывода. Некоторые микроЭВМ имеют программно-управляемый обмен данными при вводе-выводе. Однако при необходимости имеется возможность добавления в состав микроЭВМ корпуса микросхемы контроллера ПДП. В таком случае программно-управляемый обмен сохраняют для операций ввода-вывода отдельных байт (слов), которые выполняются быстрее, чем при ПДП, так как исключаются потери времени на программно-управляемую установку начальных состояний регистров и счетчиков контроллера ПДП.

### **11.3. Основные принципы построения и структуры системы ввода-вывода**

Принципы построения и структуры систем ввода-вывода сильно различаются в зависимости от типа ЭВМ и ее назначения. Определяющими факторами являются разнообразие и число периферийных устройств в составе ЭВМ, интенсивность операций ввода-вывода.

Можно выделить два характерных принципа построения и соответствующие структуры систем ввода-вывода: *ЭВМ с одним общим интерфейсом* и *ЭВМ с множеством интерфейсов и процессорами (каналами) ввода-вывода*.

*Структура с одним общим интерфейсом* (рис. 11.3) предполагает наличие общей шины (магистрали), к которой подсоединяются все модули, в совокупности образующие ЭВМ: процессор, оперативная и постоянная памяти и периферийные устройства. В каждый данный момент через общую шину может происходить обмен данными только между одной парой присоединенных к ней модулей. Таким образом, модули ЭВМ разделяют во времени один общий интерфейс, причем процессор выступает как один из модулей системы.

Периферийные устройства подсоединяются к общей шине с помощью блоков управления (контроллеров) периферийными устройствами, осуществляющих согласование форматов данных, используемых в ПУ, с форматом, принятым для передачи по общей шине («информационная шина интерфейса»). Последний

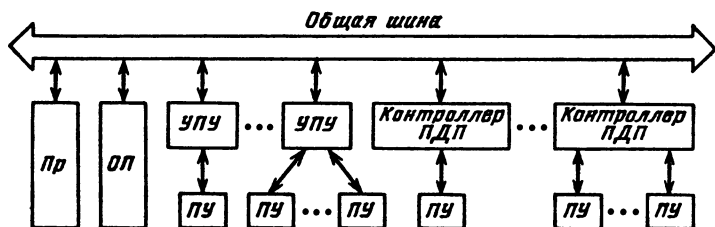


Рис. 11.3. Структура системы ввода-вывода на основе общего интерфейса

обычно соответствует машинному слову процессора и (или) ширине выборки в ОП.

Однако такой способ обмена, как было показано ранее, мало подходит для ПУ с поблочной передачей данных (ЗУ на дисках, лентах и некоторые другие ПУ). В этом случае применяют прямой доступ к памяти и контроллеры ПДП. Блоки буферизации и согласования форматов передаваемых данных сравнительно несложны, если совпадают или невелика кратность ширины выборки оперативной памяти (информационной ширины общей шины) и формата данных (обычно однобайтового), используемого для передачи в большинстве ПУ. Это имеет место в малых ЭВМ, микропроцессорах и микроЭВМ, где длина слова 1—2 байта.

Наоборот, в ЭВМ общего назначения с шириной выборки и форматом обмена между процессором и ОП 4—8 байт использование общего интерфейса потребовало бы усложнения блоков буферизации и согласования форматов в УПУ.

При общем интерфейсе аппаратура управления вводом-выводом рассредоточена по отдельным модулям и ее общий объем существенно зависит от числа ПУ в составе ЭВМ, особенно от числа ПУ с поблочным обменом, требующим контроллеров прямого доступа к памяти.

Следует отметить, что в структуре, представленной на рис. 11.3, процессор не полностью освобождается от управления операциями ввода-вывода. Более того, во время операции передачи данных интерфейс оказывается занятым, а связь процессора — с памятью заблокированной. Все это приводит к снижению производительности ЭВМ.

Сказанное объясняет, почему общий интерфейс (интерфейс с общей шиной) нашел широкое применение (приобрел даже характер стандартного архитектурного решения) применительно к малым и микроЭВМ, которые имеют короткое слово, небольшой объем периферийного оборудования и к общей производительности которых предъявляются умеренные требования. Для

машин этого класса существенным является обеспечиваемые общим интерфейсом простота реализации системы ввода-вывода и гибкость при построении различных конфигураций вычислительных комплексов.

В то же время интерфейс с общей шиной оказывается мало-пригодным для высокопроизводительных ЭВМ общего назначения, работающих с многобайтными словами, большим набором периферийных устройств, в том числе внешних ЗУ при высокой интенсивности процессов ввода-вывода данных.

Широко используются несколько модификаций общего интерфейса: «общая шина» (см. § 11.10), «мультишина» (см. § 11.11) и др.

*Структура системы ввода-вывода с процессорами (каналами) ввода-вывода.* В ЭВМ общего назначения система ввода-вывода строится путем централизации аппаратуры управления вводом-выводом на основе применения программно-управляемых процессоров ввода-вывода, иначе называемых каналами ввода-вывода (рис. 11.4). Обмен информацией между ОП и ПУ совершается через канал ввода-вывода.

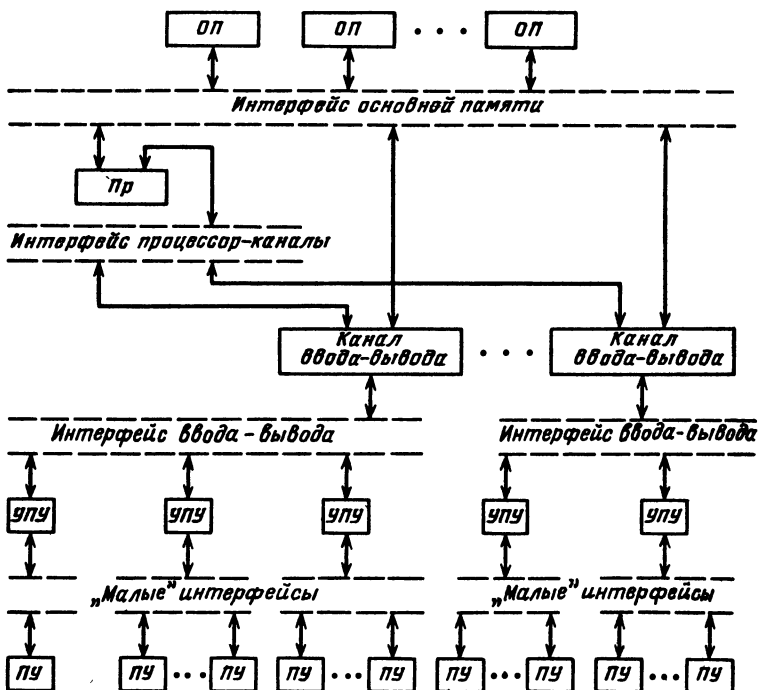


Рис. 11.4. Интерфейсы ЭВМ общего назначения (ЕС ЭВМ)

Благодаря наличию в составе ЭВМ каналов ввода-вывода, способных реализовывать достаточно сложные процедуры, появляется возможность полностью разгрузить процессор от управления операциями ввода-вывода.

В вычислительной машине с каналами ввода-вывода отсутствует однородность в структуре потоков и форматах передаваемых данных, что приводит к необходимости иметь в ЭВМ, как это показано на рис. 11.4, несколько специализированных интерфейсов.

Можно выделить четыре типа интерфейсов: интерфейс основной (оперативной) памяти, интерфейс «процессор — каналы», интерфейс ввода-вывода (интерфейс периферийных устройств), интерфейс периферийных аппаратов («малый» интерфейс). Последний имеет место и в ЭВМ с общим интерфейсом.

Через *интерфейс основной памяти* производится обмен информацией между ОП, с одной стороны, и процессором (процессорами) и каналами ввода-вывода — с другой.

*Интерфейс «процессор — каналы»* предназначается для передачи информации между процессорами и каналами ввода-вывода. Через *интерфейс ввода-вывода* происходит обмен информацией между каналами и УПУ. Через «малый» интерфейс осуществляется передача информации между УПУ и ПУ.

Наиболее быстродействующими являются интерфейс основной памяти и интерфейс «процессор — каналы». В (из) ОП информация передается словами или словами двойной длины (4—8 байт). Через интерфейс ввода-вывода информация передается байтами или парами байт. При проектировании ЭВМ стремятся унифицировать интерфейсы, в первую очередь интерфейсы, обеспечивающие сопряжение с периферийными устройствами (интерфейсы ввода-вывода).

Интерфейсы периферийных аппаратов не могут быть унифицированы, так как сами периферийные аппараты весьма разнообразны по принципу действия, по выполняемым операциям и по используемым форматам данных и сигналам.

Интерфейсы характеризуются следующими параметрами:

*пропускной способностью интерфейса* — количеством информации, которая может быть передана через интерфейс в единицу времени;

*максимальной частотой передачи информационных сигналов* через интерфейс;

*максимально допустимым расстоянием* между соединяемыми устройствами;

*динамическими параметрами интерфейса* — временем передачи отдельного слова и блока данных с учетом продолжительности процедур подготовки и завершения передачи;

*общим числом проводов (линий) в интерфейсе;  
информационной шириной интерфейса — числом бит или  
байт данных, передаваемых параллельно через интерфейс.*

#### **11.4. Основные функции каналов ввода-вывода.**

##### **Управляющая информация для операций ввода-вывода**

При определении функций, которые следует возложить на каналы ввода-вывода, нужно исходить из необходимости обеспечения условия для реализации параллельной во времени работы процессора над программой с выполнением ПУ операций ввода-вывода. Для этого надо в возможно большей степени освободить процессор от управления операциями обмена информацией между периферийными устройствами и ОП, возложить это на каналы, управляемые *канальными программами*.

За процессором следует оставить лишь инициирование операции ввода-вывода, задание номеров канала и ПУ, участвующих в операции, и указание адреса начала программы канала для задаваемой операции ввода-вывода.

Канал призван обеспечивать прямой доступ к памяти, а поэтому подобно рассмотренному в § 11.3 контроллеру ПДП должен выполнять следующие функции: задание размера массива данных и области памяти, участвующих в обмене информацией, формирование адресов последовательных ячеек ОП, используемых в передаче, подсчет числа единиц данных (слов, байт и т. д.), прошедших через канал, и определение момента завершения передачи массива данных. При этом канал должен осуществлять буферизацию и преобразование форматов передаваемых данных для согласования работы ОП и ПУ.

Помимо указанных функций на канал возлагается ряд дополнительных для минимизации участия процессора в операциях ввода-вывода.

*Организация цепочки данных.* Возможны случаи, когда массив информации, предназначенный для операции ввода или вывода с некоторым ПУ, не располагается в памяти подряд, а состоит из нескольких подмассивов, размещенных в произвольно расположенных участках ОП. Чтобы в этом случае ввод (вывод) каждого подмассива не требовал включения в программу процессора отдельной команды ввода-вывода, а передача всех подмассивов иницировалась всего одной командой процессора, канал должен допускать задание в канальной программе **цепочки данных** для передачи такого составного массива.

*Организация пропуска информации.* При операциях ввода может возникнуть необходимость переносить в память с носителя информации отдельные части массива, пропуская ненужные

данные. Должна иметься возможность задания в канальной программе пропуска информации в цепочке данных и реализация пропуска без привлечения процессора для выполнения этой процедуры.

*Организация цепочки операций.* Для выполнения программы может оказаться необходимым такой обмен информацией между ОП и некоторым ПУ, для выполнения которого с этим ПУ должна выполняться определенная последовательность операций ввода-вывода. Например, при работе с ЗУ на магнитных дисках может потребоваться следующая последовательность операций: а) установить головки на  $i$ -й цилиндр; б) прочитать информацию с  $j$ -й поверхности дисков; в) прочитать информацию с  $(j+1)$ -й поверхности; г) установить головки на  $k$ -й цилиндр и т. д. Целесообразно, чтобы при подобных последовательностях операций ввода-вывода с одним и тем же ПУ не требовалось для каждой новой операции участия процессора, т. е. не требовалась новая команда ввода-вывода в программе процессора. Для этого канал ввода-вывода должен допускать задание цепочек операций в программе канала.

*Блокировка контроля неправильной длины* считанного массива. Операции ввода-вывода сопровождаются автоматическим контролем, в том числе контролем соответствия длины массива, считанного или записанного в результате операции ввода-вывода, длине физической записи. При нарушении соответствия возникает прерывание от ввода-вывода по *неправильной длине*. Поскольку имеется много случаев, когда следует блокировать это прерывание (соответствующие примеры будут приведены далее), целесообразно, чтобы канал допускал задание такой блокировки в канальной программе.

*Формирование запросов прерывания от ввода-вывода.* Канал должен прерыванием извещать процессор об окончании каждой операции ввода-вывода, а также об обнаружении ошибки или каких-либо других необычных условий, вследствие чего произошло принудительное окончание выполняемой операции. Это прерывание формируется автоматически аппаратурой канала. Наряду с этим должна иметься возможность задания в программе канала прерывания на любом этапе операции ввода-вывода. Такое прерывание называется *программно-управляемым*. Оно не нарушает нормальное выполнение текущей операции ввода-вывода. Появление запроса программно-управляемого прерывания означает, что выполнены все операции ввода-вывода, предшествующие в канальной программе этому запросу. Это позволяет процессору следить за выполнением канальной программы ввода-вывода и начинать обработку данных сразу после выполнения очередного этапа операции ввода-вывода.



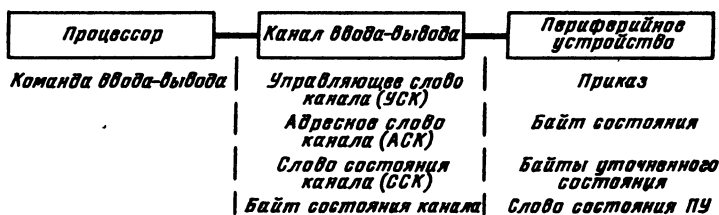


Рис. 11.5. Иерархия управляющей информации для операции ввода-вывода (ЕС ЭВМ)

*Управляющая информация для операций ввода-вывода.* В ЭВМ с каналами ввода-вывода управление вводом-выводом строится иерархическим образом (рис. 11.5). В операциях ввода-вывода участвуют три типа устройств: процессор (первый уровень управления), канал ввода-вывода (второй уровень), периферийное устройство (третий уровень). Каждому типу устройств соответствует определенный вид управляющей информации: процессору — *команды ввода-вывода*, каналу — *управляющие слова канала (УСК)*, периферийному устройству — *приказы*. Кроме того, в управлении вводом-выводом используются коды состояния канала [*слово состояния канала (ССК)*] и ПУ (*байт состояния* и *байты уточненного состояния*). О кодах состояния см. в § 11.10.

Форматы основных видов управляющей информации для операций ввода-вывода в ЕС ЭВМ представлены на рис. 11.6.

Каждая операция или совокупность операций ввода-вывода производится под управлением соответствующей программы канала, реализуемой аппаратурными средствами канала, ПУ (и его УПУ) и интерфейса ввода-вывода.

Программа канала представляет собой некоторую последовательность УСК, обеспечивающую выполнение определенной операции ввода-вывода. Обычно каналные программы хранятся в ОП.

Таким образом, в ЭВМ организуется параллельное выполнение во времени взаимодействующих между собой программно-управляемых процессов: выполнение процессором программы обработки данных и выполнение каналами и ПУ канальных программ операций ввода-вывода.

Команды ввода-вывода являются привилегированными и выполняются только в состоянии «супервизор». Все команды ввода-вывода завершаются формированием признака результата в ССП.

Система команд ЭВМ содержит небольшое число универсальных по отношению к разным типам ПУ команд ввода-вывода.

да. Так, в ЕС ЭВМ I очереди для управления вводом-выводом использовались четыре команды: «Начать ввод-вывод», «Остановить ввод-вывод», «Проверить ввод-вывод», «Проверить канал».

**Команда «Начать ввод-вывод»** инициирует одну или несколько (цепочку) операций ввода-вывода с указанным в команде каналом и ПУ. Этой командой процессора начинаются операции ввода-вывода с любым ПУ. О завершении отдельного ее этапа, или цепочки операций, канал сообщает процессору путем прерываний.

Во всех указанных выше командах ввода-вывода (кроме первой) достаточно указывать код операции и номер канала и ПУ. В команде «Начать ввод-вывод», кроме того, нужно указывать адрес первого УСК в программе канала. Для единообразного представления всех команд ввода-вывода в ЕС ЭВМ для них принят один общий формат, показанный на рис. 11.6, а. Адрес первого УСК программы канала содержится в так называемом *адресном слове канала* (АСК) (рис. 11.6, б), хранимом в определенной ячейке ОП, куда оно должно быть помещено до начала выполнения команды «Начать ввод-вывод».

При выполнении команды ввода-вывода содержимое указанного в команде регистра  $B_1$  складывается с числом в поле  $D_1$  и полученная сумма располагается в разрядах 21—31 слова, при этом разряды 21—23 и 24—31 указывают соответственно номер канала и номер ПУ.

Адресное слово канала содержит код «ключа», используемый для защиты памяти (см. гл. 14) при данной операции ввода-вывода. Признак результата, формируемый при выполнении команды «Начать ввод-вывод», указывает, нормально ли прошел пуск операции ввода-вывода.

По команде «Остановить ввод-вывод» операция ввода-вывода может быть принудительно прекращена процессором до ее завершения в адресуемых командой канале и ПУ.

Команды «Проверить канал» и «Проверить ввод-вывод» позволяют процессору определять состояние канала и ПУ. По команде «Проверить канал» канал в фиксированной ячейке ОП формирует ССК (рис. 11.6, в) и устанавливает в ССП признак результата, определяющий состояние адресуемого канала: канал доступен, канал хранит условия прерывания, канал работает в монопольном режиме, канал выключен. При выполнении команды «Проверить ввод-вывод» из ПУ в канал выдается байт его состояния, а из канала в процессор поступает признак результата, указывающий следующие возможные ситуации: адресуемое ПУ доступно, ССК записано, канал (подканал) занят, адресуемое ПУ выключено.

Дополнительные команды ввода-вывода, введенные в ЕС ЭВМ II очереди, приведены в § 11.8.

**Управляющее слово канала (УСК).** В ЕС ЭВМ формат УСК соответствует двойному слову, т.е. 64 разрядам (рис. 11.6, г). При выполнении канальной программы ее управляющие слова выбираются последовательно из памяти, если только не предусмотрено программой канала нарушение естественного порядка выборки УСК. Для удобства дальнейшего изложения материала будем пользоваться следующей упрощенной структурой УСК:

Код операции (приказ)	Указатели					Адрес данных (АД)	Счетчик данных (СчД)
	ЦД	ЦО	УБ	ПИ	ПР		

*Код операции (приказ)* определяет как для канала, так и для ПУ тип операции, задаваемой этим управляющим словом.

Приказ представляет собой часть УСК, которая задает операцию, выполняемую каналом, каналом и ПУ совместно или одним ПУ. В последних двух случаях приказ передается в ПУ и инициирует в нем определенные действия (например, установку головок на нужный цилиндр в ЗУ на дисках, запись информации и др.). Приказ содержит информацию, специфичную для данного типа ПУ.

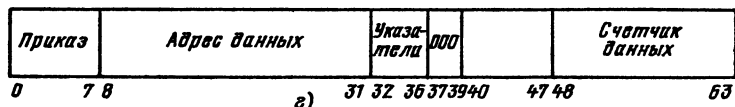
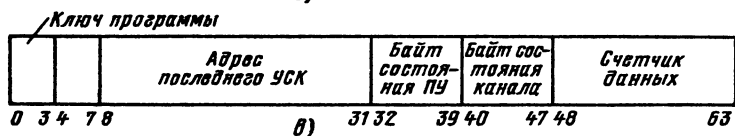
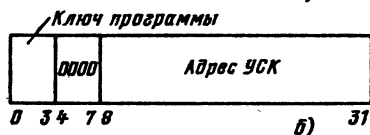
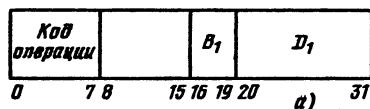


Рис. 11.6. Форматы команды ввода-вывода (а), адресного слова канала (б), слова состояния канала (в) и управляющего слова канала (г) в ЕС ЭВМ

Приведем основные виды приказов, допускающих ряд модификаций:

«Записать» (вывод информации из ОП в ПУ);

«Прочитать» (ввод информации из ПУ в ОП);

«Прочитать в обратном направлении» (только для ЗУ на магнитных лентах);

«Управление» (перемещение магнитной ленты, перемещение головок в ЗУ на дисках и другие служебные операции);

«Проверить ввод-вывод» (запрос байта состояния);

«Уточнить состояние» (запрос кода уточненного состояния ПУ).

Кроме операций, управляющих передачами информации, предусматриваются также операции, которые нужны для построения самих программ канала. К ним относится так называемый «Переход в канале». Эта операция является внутренней процедурой канала и непосредственно на работу ПУ не влияет. Управляющее слово, задающее переход в канале, указывает адрес следующего УСК в цепи и, таким образом, позволяет выполнять программы канала, в которых УСК располагаются не в последовательных ячейках памяти. Этот вид перехода в программе канала является безусловным переходом.

Для реализации ветвлений в программах канала применяются условные переходы. В зависимости от признаков, характеризующих состояние ПУ, после выполнения очередного управляющего слова следующее УСК в цепи операции либо выполняется, либо пропускается.

В ЕС ЭВМ признаком условного перехода в канальной программе служит содержимое разряда «Модификатор состояния» в байте состояния, который ПУ посылает в канал по завершении операции, предписанной текущим УСК. Если модификатор состояния содержит 0, то выбирается и выполняется следующее по порядку УСК в цепочке операций. Если модификатор равен 1, следующее УСК пропускается.

Далее будет рассмотрен пример программы канала, в которой используются условный и безусловный переходы.

Адрес данных (АД) и счетчик данных (СчД) определяют область памяти, используемую в операции ввода-вывода. Адрес данных указывает адрес первого (или последнего при обратном вводе) байта из массива информации; счетчик данных указывает число байт.

*Указатель цепочки данных (ЦД).* При  $ЦД=0$  операция после использования данного УСК оканчивается, при  $ЦД=1$  она продолжается с новым массивом данных, указанным в следующем УСК.

*Указатель цепочки операций (ЦО).* При  $ЦО=0$  программа

канала для данного ПУ заканчивается на рассматриваемом УСК, если нет указаний о продолжении цепочки данных. Если ЦД=0, то при ЦО=1 после выполнения действий, предусмотренных данным УСК, следующее по порядку УСК выбирается из памяти и начинается выполнение новой операции ввода-вывода с тем же ПУ.

*Указатель блокировки (УБ)* сигнала неправильной длины, формирующегося (в отсутствие признака цепочки данных ЦД=1) при несоответствии числа фактически переданных байт длине физической записи. Этот сигнал прекращает выполнение канальной программы и вызывает прерывание программы процессора. При УБ=1 сигнал неправильной длины блокируется.

*Указатель пропуска информации (ПИ)*. При ПИ=1 передача информации между каналом и оперативной памятью подавляется, канал осуществляет только подсчет слов, проходящих между каналом и ПУ.

*Указатель прерывания (ПР)*. При ПР=1 канал посылает в процессор запрос программно-управляемого прерывания.

### **Примеры программ канала**

**Пример 1.** Выборочная передача информации между ПУ и ОП.

Предположим, что в ОП машины должны быть приняты из некоторой зоны магнитной ленты, содержащей 500 байт, первые 10 и последние 20 байт. Если считать, что требуемая зона уже подведена под считывающие головки, то для выполнения данной операции необходимо выполнить следующую последовательность УСК, образующих цепь данных:

Адрес УСК	Приказ	Указатель					Адрес данных (АД)	Счетчик данных (СчД)
		ЦД	ЦО	УБ	ПИ	ПР		
<i>l</i>	«Прочитать»	1	0	0	0	0	$\alpha$	10
<i>l+8</i>	«Прочитать»	1	0	0	1	1	0	470
<i>l+16</i>	«Прочитать»	0	0	1	0	0	$\alpha+10$	20

**Примечание.** В УСК, продолжающих цепочку данных, в качестве приказа может быть записан любой код, кроме кода «переход в канале». Обычно используют нулевой код или повторяют приказ из первого УСК цепочки данных.

Первое УСК выполняет ввод первых 10 байт в ячейки с адресом от  $\alpha$  до  $\alpha+9$ . Второе УСК обеспечивает пропуск следующих 470 байт; установленный в этом УСК указатель ПР=1, вызывая программно-управляемое прерывание, позволяет программе не-

медленно начать обработку начальных 10 байт. Третье УСК выполняет ввод в ОП последних 20 байт из зоны магнитной ленты и заканчивает программу канала, так как ЦД=0 и ЦО=0. Автоматически формируемое каналом по окончании цепочки УСК прерывание сигнализирует процессору, что затребованная операция ввода-вывода выполнена полностью.

**Пример 2.** Ввод данных с перфокарт.

Пусть необходимо произвести ввод данных с двух перфокарт: с первой 25 байт, расположенных на карте в колонках с 31-й по 55-ю, а со второй 30 байт из колонок 41—70 и расположить их в ОП, начиная с адресов соответственно  $\alpha$  и  $\beta$ . Для рассматриваемого примера приведем канальную программу:

Адрес УСК	Приказ	ЦД	ЦО	УБ	ПИ	ПР	АД	СчД
$l$	«Прочитать»	1	0	0	1	0	0	30
$l+8$	«Прочитать»	0	1	1	0	0	$\alpha$	25
$l+16$	«Прочитать»	1	0	0	1	1	0	40
$l+24$	«Прочитать»	0	0	1	0	0	$\beta$	30

В программе канала организуется цепочка операций, так как считывание каждой перфокарты составляет самостоятельную операцию ввода-вывода. Использование указателя блокировки неправильной длины позволяет сократить число УСК в программе и освободить канал раньше, чем закончится чтение физической записи (в данном случае перфокарты).

**Пример 3.** Поиск информации в ЗУ на магнитных дисках (ЗУД).

Пусть необходимо найти в ЗУД по ключу (см. гл. 5) блок данных, пользуясь ключом-эталоном, имеющим длину  $n$  байт и расположенных в ОП, начиная с адреса  $\alpha$ , а затем первые  $m$  байт передать в ОП, начиная с адреса  $\beta$ .

Блок управления ЗУД, просматривая последовательно все блоки информации на носителе (или заданной дорожке), должен найти блок, ключ которого совпадает с эталоном. Поскольку поле ключа может иметь большую длину (до 256 байт), невыгодно устанавливать в блоке управления ЗУД триггерные регистры для хранения ключа-эталона целиком на все время поиска. Ключ-эталон многократно считывается из ОП (при прохождении каждого блока информации под головкой) и побайтно сравнивается с данными, считываемыми с диска.

Для организации такого поиска построим программу канала в виде цепи операций с использованием безусловного перехода, выполняемого операцией «Переход в канале», и условного пере-

хода по признаку «Модификатор состояния» в байте состояния, который посылает в канал ЗУД.

Предполагая, что требуемые цилиндр и дорожки в ЗУД уже выбраны, можно написать следующую программу канала:

Адрес УСК	Приказ	ЦД	ЦО	УБ	ПИ	ПР	АД	СчД
$l$	«Поиск по ключу»	0	1	1	0	0	$\alpha$	$n$
$l+8$	«Переход в канале»	0	1	0	0	0	$l$	0
$l+16$	«Прочитать»	0	0	1	0	0	$\beta$	$m$

Первое УСК программы канала инициирует в ЗУД операцию поиска. Запоминающее устройство на диске выбирает из первого подошедшего под головку блока информации байты ключа и последовательно сравнивает их с  $n$  байтами эталона, которые выбираются каналом из ОП, начиная с адреса  $\alpha$ . Если совпадения ключей нет, то следующее УСК (*переход в канале*) вызывает повторное выполнение УСК из ячейки  $l$ , т. е. повторение поиска для следующего на дорожке блока информации. Если блок информации с требуемым полем ключа найден, то ЗУД формирует байт состояния с 1 в разряде «Модификатора состояния». Канал пропускает в своей программе УСК из ячейки  $l+8$  и переходит к выполнению УСК из ячейки  $l+16$ . В этом случае в ОП вводятся  $m$  байт, которые помещаются в группу последовательных ячеек, начиная с ячейки  $\beta$ .

## 11.5. Основные типы и структуры каналов ввода-вывода

Способ организации взаимодействия ПУ с каналом определяется соотношением быстродействия ОП и ПУ. По этому признаку ПУ можно классифицировать на две группы: быстродействующие (ЗУ на барабанах, дисках, лентах и другие устройства) со скоростью приема и выдачи информации примерно  $(0,1-10)10^6$  байт/с и медленнодействующие (перфоленточные и перфокарточные устройства, печатающие устройства и др.) со скоростью около 1—2 тыс. байт/с и менее. Оперативная память может выдавать или принимать данные со скоростью примерно до  $10 \cdot 10^6$  байт/с.

В зависимости от соотношения быстродействия ОП и ПУ в каналах ввода-вывода реализуются два режима работы: монопольный и разделения времени (мультиплексирования).

**Монопольный режим.** После установления связи между каналом и ПУ последнее монополизирует канал на все время, пока полностью не завершится инициированная процессором

канальная программа (цепочка операций) работы с данным ПУ и не будут произведены все предусмотренные этой программой передачи данных между ПУ и ОП. На все время выполнения данной канальной программы канал оказывается занятым для других ПУ.

*Режим разделения времени (режим мультиплексирования).* Несколько ПУ разделяет во времени канал ввода-вывода, при этом каждое из параллельно работающих с данным каналом ПУ связывается с каналом на короткие промежутки времени только после того, как ПУ подготовлено к приему или выдаче очередной порции информации (байта, группы байт и т. п.). Промежуток времени, в течение которого происходит передача информации между каналом и подготовленным к этому ПУ, может быть назван *сеансом связи*. Сеансы связи различных ПУ чередуются между собой. Во время сеанса связи одного из устройств с каналом другие устройства могут выполнять работу, не требующую использования средств канала (например, печатать очередной символ или считывать символ с носителя информации).

В соответствии с преимущественно реализуемым режимом работы различают каналы ввода-вывода *мультиплексный*, осуществляющий мультиплексирование ПУ, и *селекторный*, взаимодействующий с ПУ в монопольном режиме.

*Мультиплексный (байт-мультиплексный) канал* одновременно обслуживает несколько параллельно работающих ПУ, попеременно организуя с ними сеансы связи для передачи между ОП и ПУ небольших порций информации (1 байта или нескольких).

Если несколько ПУ подготовилось к очередному сеансу связи и запрашивает обслуживание со стороны мультиплексного канала, то канал выбирает одно из них в соответствии с принятыми для данной системы приоритетными правилами, например, в соответствии с порядком подключения устройств к каналу. Остальные устройства, готовые к сеансу связи, должны ожидать, когда подойдет их очередь на обслуживание. Мультиплексный канал предназначен главным образом для работы со сравнительно медленными устройствами, способными ожидать обслуживания без потери информации. Аппаратурные средства мультиплексного канала можно условно разделить на две части (рис. 11.7, а): средства, предназначенные для обслуживания отдельных ПУ, присоединенных к каналу, и оборудование, являющееся общим для всех устройств и разделяемое всеми устройствами во времени.

Средства канала, выделенные для обслуживания отдельных устройств, принято именовать *подканалом*.



Число подканалов определяет максимальное число одновременно работающих с данным каналом ПУ. Физически подканал реализуется в виде участка памяти, в котором хранятся параметры операции ввода-вывода, выполняемой данным устройством: текущие значения адреса и счетчика данных, код и указатели операции ввода-вывода, адрес следующего УСК и др. В качестве памяти для хранения этих параметров может использоваться либо специальная память, встроенная в мультимплексный канал, либо участок ОП машины.

Общее оборудование мультимплексного канала представляет собой набор триггерных регистров и комбинационных схем, позволяющих осуществлять обмен информацией между ОП и ПУ, модификацию текущих параметров операции ввода-вывода.

*Селекторный канал* предназначается для монопольного обслуживания одного ПУ. При работе с селекторным каналом ПУ после пуска операции остается связанным с каналом до окончания цепочки операций. До завершения цепочки операций селекторный канал по отношению к процессору представляется занятым устройством.

Управляющее слово, выбранное селекторным каналом из памяти, содержится до окончания всех предписанных им действий в триггерных регистрах канала. Необходимые изменения текущих параметров операции производятся быстро с помощью соответствующих действий над содержимым триггерных регистров. Таким образом, все средства селекторного канала монополизируются на время операции одним ПУ. Можно счи-

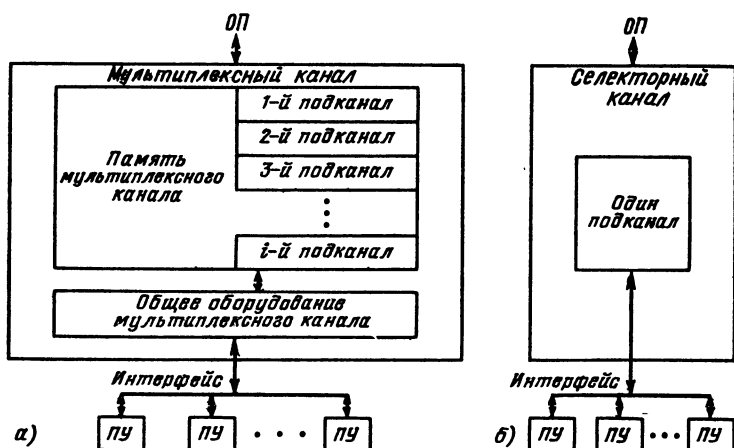


Рис. 11.7. Типы каналов:  
а — мультимплексный; б — селекторный

тать, что селекторный канал содержит только один подканал (рис. 11.7, б).

Вследствие отсутствия потерь времени на перезапоминание текущих параметров операций ввода-вывода селекторный канал обладает высокой степенью готовности к обслуживанию пущенного им устройства и предназначается для работы с быстродействующими устройствами, которые могут терять информацию вследствие задержек в обслуживании (ЗУ на магнитных лентах, дисках и др.).

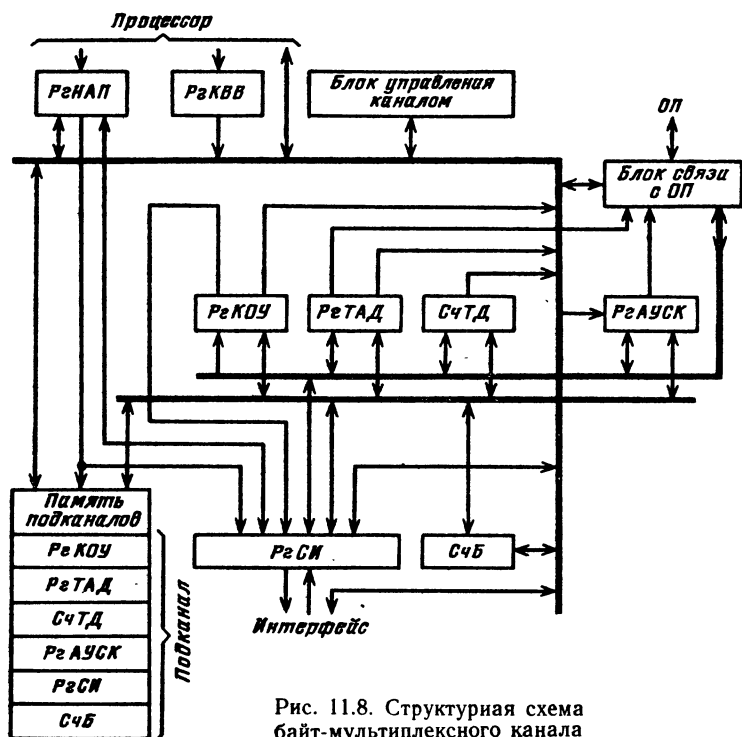
Возможность монопольного обслуживания ПУ вводится иногда и в мультиплексный канал. В этом режиме один из подканалов полностью занимает канал.

*Блок-мультиплексные каналы* позволяют осуществлять параллельную работу нескольких ВЗУ с прямым доступом. Операции, не связанные с передачей данных (установка головок на цилиндр, поиск записи и др.), выполняются для нескольких устройств в мультиплексном режиме, а передача блока информации происходит в монопольном (селекторном) режиме. Важным свойством блок-мультиплексного канала является возможность мультиплексирования передач блоков данных, относящихся к различным ВЗУ прямого доступа (см. § 11.9). Блок-мультиплексный канал содержит несколько подканалов.

*Структура байт-мультиплексного канала* (рис. 11.8). В состав байт-мультиплексного канала входят следующие основные узлы:

1) набор триггерных регистров, в которых во время сеанса связи с ПУ содержатся текущие параметры активного подканала (активным называется тот подканал, для которого производится данный сеанс связи). Он состоит из регистра кода операции и указателей *РгКОУ*, регистра текущего адреса данных *РгТАД*, содержащего адрес байта в ОП, участвующего в данный момент в операции ввода-вывода, счетчика текущих данных *СчТД*, указывающего число байт, которое осталось ввести в ОП или вывести в данной операции, регистра адреса управляющего слова канала *РгАУСК*, определяющего адрес очередного УСК в цепи управляющих слов;

2) память подканалов, представляющая собой внутреннее ЗУ канала и предназначенная для хранения текущих параметров операций ввода-вывода, относящихся к пассивным подканалам. Каждому подканалу отведен в памяти участок, в котором хранится содержимое регистров *РгКОУ*, *РгТАД*, *СчТД*, *РгАУСК*, соответствующее последнему сеансу связи для данного подканала. Кроме того, в памяти подканалов предусматривается место для хранения некоторых других параметров операции, о которых будет сказано далее. Подканалы в памяти располагаются упорядоченно по возрастанию номера подканала, который,



таким образом, может использоваться для адресации ячеек памяти подканалов. При формировании адресов ячеек памяти подканалов используется содержимое регистра номера активного подканала *РгНАП*;

3) регистр связи с интерфейсом *РгСИ*, в который поступает информация, получаемая из ПУ при вводе и из которого в ПУ выдается выводимая информация.

Обмен информацией канала с ОП производится словами или двойными словами. Обмен с ПУ производится обычно более мелкими единицами информации, например байтами. Поэтому при выводе информации канал производит в *РгСИ* компоновку слова из поступающих в канал байт, а при выводе — развертку слова в *РгСИ* в последовательность выдаваемых в ПУ байт. Для определения конца компоновки или развертки слова используется счетчик байт *СчБ*, который указывает номер последнего обработанного байта в текущем слове данных. Содержимое *СчБ* и *РгСИ* запоминается в соответствующем участке памяти подканалов наряду с регистрами *РгКОУ*, *РгТАД* и т. д.;

4) регистр команды ввода-вывода *РзКВВ*, предназначенный для хранения кода (двухбитового) операции команды ввода-вывода, поступающего в канал из процессора, когда процессор в соответствии со своей программой выполняет новую команду ввода-вывода.

Выполнение операции ввода-вывода в канале можно рассматривать как совокупность нескольких видов процедур, из которых наиболее важными являются две: *начальная выборка* и *обслуживание* ПУ. Начальная выборка производится по инициативе процессора при пуске новой операции ввода-вывода. Процессор передает в канал код операции команды ввода-вывода, номер ПУ и адрес первого УСК. Эти параметры заносятся каналом в регистры *РзКВВ*, *РзНАП* и *РзАУСК*. В тех машинах, где адрес первого УСК не указывается в команде процессора, а содержится в специализированной ячейке ОП (в адресном слове канала), канал, заполнив *РзКВВ* и *РзНАП*, сам считывает из ОП адрес первого УСК и заносит его в *РзАУСК*. После этого, используя содержимое *РзАУСК* как адрес, канал выбирает из ОП первое УСК, размещая его поля в регистрах *РзКОУ*, *РзТАД*, *СчТД*. К содержимому *РзАУСК* прибавляется число, равное длине управляющего слова, после чего *РзАУСК* указывает адрес следующего УСК в цепочке. Канал, устанавливая признак результата  $ПР=00$ , запускает в работу требуемое ПУ. Если устройство свободно и во время начальной выборки не обнаружены программные или аппаратные ошибки, считается, что пуск произошел нормально. Канал, устанавливая признак результата  $ПР=00$ , сообщает об этом процессору, который переходит к выполнению следующей команды своей программы. Канал записывает в участок памяти подканалов, соответствующий номеру ПУ в *РзНАП*, содержимое регистров *РзКОУ*, *РзТАД*, *СчТД*, *РзАУСК*, *СчБ*, *РзСИ*. На этом начальная выборка заканчивается, канал освобождается и готов к обслуживанию ранее пущенных ПУ или приему новых команд из процессора.

Процедура обслуживания производится по инициативе ПУ, которое посылает в канал требование обслуживания, после того как устройство готово к передаче (приему) очередной порции информации (например, байта). Канал, восприняв это требование, получает от интерфейса номер ПУ, пересылает его через *РзСИ* в *РзНАП* и в соответствии с содержимым *РзНАП* считывает из памяти подканалов участок, соответствующий активному подканалу. Текущие параметры операции из памяти подканалов размещаются в регистрах *РзКОУ*, *РзТАД*, *СчТД*, вводимое (или выводимое) слово — в регистре *РзСИ*, номер текущего байта — в *СчБ*, адрес следующего УСК — в *РзАУСК*.

При операции ввода из ПУ поступает байт данных, который размещается в *РзСИ* на месте, определяемом счетчиком байт. При этом содержимое *СчБ*, *РзТАД* увеличивается, а содержимое *СчТД* (и *РзТАД* при считывании при обратном движении ленты) уменьшается на 1.

Если *СчБ* указывает, что компоновка (или развертка) слова данных окончена, то канал осуществляет связь с ОП и в соответствии с адресом,

размещенным в *РгТАД*, записывает в ОП введенное в *РгСИ* слово или считывает из ОП в *РгСИ*, а также содержимое *РгКОУ*, *РгАУСК* новое слово для вывода.

Если содержимое счетчика данных в *СчТД* не равно 0, то канал запоминает в памяти подканалов новые значения параметров из *РгТАД*, *СчТД*, *СчБ*, *РгСИ* и заканчивает сеанс связи с ПУ.

Если содержимое счетчика данных равно 0, то канал информирует ПУ об окончании операции (если только в УСК не указана цепь данных), затем, используя содержащийся в *РгАУСК* адрес следующего УСК для данного подканала, по адресу из *РгАУСК* считывает в свои регистры новое УСК и выполняет процедуру начальной выборки для пуска следующей операции в цепи управляющих слов. При обработке последнего УСК в цепочке канал посылает в процессор прерывание, сигнализирующее об окончании цепочки УСК.

*Структура селекторного канала* (рис. 11.9) содержит набор триггерных регистров, большинство из которых по своим функциям аналогично соответствующим регистрам в описанном ранее примере байтмультиплексного канала.

Регистр номера периферийного устройства *РгНПУ*, заполняемый процессором при начальной выборке, указывает, с каким из периферийных устройств проводится текущая операция. Регистр данных *РгД* и регистр предварительного управляющего слова канала *РгПУСК* являются буферными. Буферным является также регистр *РгСИ*. Благодаря этим буферным регистрам становится возможным совмещение во времени передачи с высокой скоростью данных из ПУ в ОП (или из ОП в ПУ) с выборкой из ОП следующего управляющего слова в цепочке данных или операций.

Процедуры работы селекторного канала во многом похожи на процедуры байтмультиплексного канала. Основное отличие состоит в том, что текущие параметры операции в селекторном канале в течение всей операции содержатся и модифицируются в триггерных регистрах, что обеспечивает высокое быстродействие этого канала. Кроме того, за счет введения дополнительных буферных регистров — регистра данных *РгД* и регистра предварительной выборки управляющего слова *РгПУСК* — в селекторном канале обеспечивается возможность совмещения во времени обмена информацией канала с ПУ и ОП.

При вводе информации поступающие из ПУ данные компонуются в регистре *РгСИ* в слово. Как только слово данных сформировано, оно передается в *РгД* и канал начинает связываться с ОП для записи слова в памяти. Параллельно с обращением к ОП канал может производить накопление байт следующего слова в *РгСИ*.

Аналогично совмещается связь с ОП и ПУ при выводе. Канал засылает в *РгСИ* через *РгД* очередное слово, выбранное из ОП, и пока это

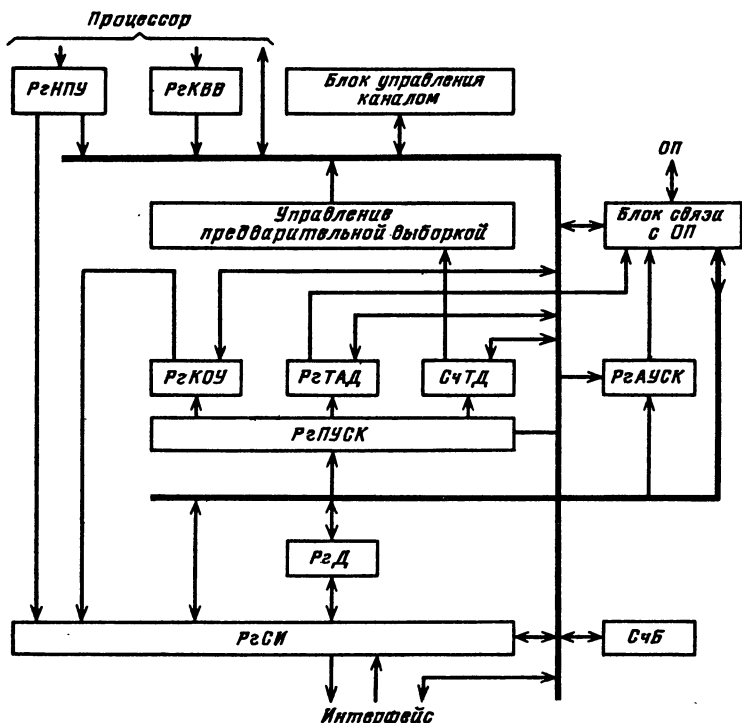


Рис. 11.9. Структурная схема селекторного канала

слово разворачивается в последовательность байт, посылаемых в ПУ, канал связывается с ОП для выборки следующего слова.

При работе канала с быстродействующими устройствами с движущимся носителем информации (магнитные ленты, диски, барабаны) имеется опасность потери информации в моменты выборки очередных управляющих слов. Примером может служить процедура исполнения цепочки данных, когда канал, не замедляя темпа поступления информации из интерфейса, должен успеть не только принимать порции данных, компоновать их в слово и записывать в память, но и выбирать следующее управляющее слово в цепочке.

В связи с этим в селекторных каналах применяют предварительную выборку управляющего слова. Схема управления предварительной выборкой следит за состоянием счетчика текущих данных, и, когда содержимое *СчТД* станет меньше принятого для данной машины значения, канал производит обращение к памяти по адресу из *РгАУСК*. Новое управляющее слово посылается в *РгПУСК*, в то время как канал продолжает обработку предыдущего УСК. После того как содержимое *СчТД*

станет равным 0, содержимое *РзПУСК* замещает старое содержимое *РзТАД*, *СчТД* и указателей в регистре *РзКОУ*.

В блоках управления каналами используют оба известных принципа построения управляющих автоматов: на основе хранимой в памяти логики (микропрограммное управление) — преимущественно в ЭВМ малой и средней производительности и на основе схемной (жесткой) логики, а также на основе комбинации обоих принципов — в высокопроизводительных ЭВМ.

В заключение данного параграфа в качестве примера приведены характеристики каналов ввода-вывода ЭВМ ЕС-1046. В состав этой машины входят два байт-мультиплексных и четыре блок-мультиплексных канала, могущих работать и как селекторные. Пропускная способность байт-мультиплексного канала 50 кбайт/с, при работе в монопольном режиме 160 кбайт/с. Пропускная способность блок-мультиплексного канала в зависимости от его номера 0,6—1,5 или 0,6—3,0 Мбайт/с с одно- или двухбайтовым интерфейсом соответственно. Общая пропускная способность при шести каналах достигает 10 Мбайт/с.

## 11.6. Буферы данных в системах ввода-вывода

Из предыдущего параграфа следует, что в организации систем ввода-вывода важное место занимают *буферы данных* (буферные ЗУ или регистры), располагаемые между периферийными устройствами и ядром ЭВМ (основной памятью, каналами ввода-вывода), а иногда между каналами и ОП. Так, например, в представленных на рис. 11.8 и 11.9 структурах каналов буферами данных являются регистры *РзСИ*, *РзД*, *РзПУСК*. Поэтому целесообразно более подробно рассмотреть общие вопросы использования буферов в системах ввода-вывода [29].

Буферы данных выполняют следующие функции:

- согласование форматов данных, с которыми работают передающее и принимающее информационное устройство;

- согласование скоростей работы передающего и принимающего устройств;

- виртуальное (кажущееся) изменение количественных и качественных характеристик периферийного устройства (относительно устройств ядра ЭВМ).

К характеристикам ПУ, существенным для процесса обмена информацией, отнесем:

- формат  $\Phi_{\text{ПУ}}$  единиц информации (байт, слово и т. д.), которые передает или принимает ПУ;

- интервал времени  $T_{\text{ПУ}}$  между последовательно передаваемыми единицами информации, и интервал  $\Theta_{\text{ПУ}}$  между запросами ПУ на обслуживание (прием или выдачу единицы информации)

Рис. 11.10. Буфер данных в системе ввода-вывода

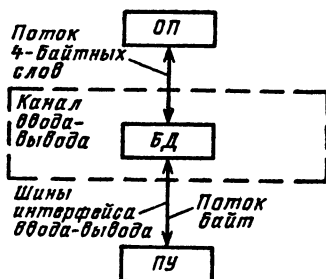
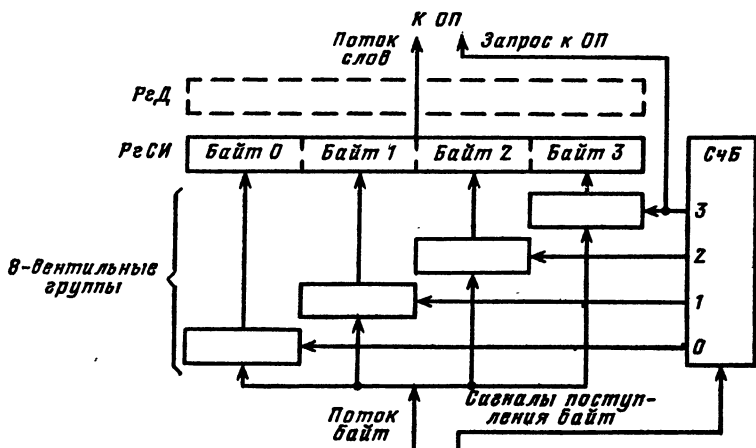


Рис. 11.11. Структура буфера данных ( $k_\Phi=4$ )



со стороны ОП или другого устройства ядра ЭВМ (для ПУ с постоянной скоростью передачи данных);

максимально допустимое время ожидания  $\tau_{\text{ПУ}}$  периферийным устройством обслуживания со стороны ядра ЭВМ (приема информации в ОП и т. п.). Можно считать  $\theta_{\text{ПУ}} \approx T_{\text{ПУ}}$  и  $\tau_{\text{ПУ}} \approx T_{\text{ПУ}}$ .

На рис. 11.10 представлен буфер данных БД, связывающий ПУ (точнее, его УПУ) с ОП (ядром ЭВМ). Независимо от направления передачи данных будем сторону буфера, обращенную к ядру ЭВМ, называть внутренней, а обращенную к ПУ — внешней.

Буфер выполняет преобразование форматов данных с коэффициентом преобразования, который с учетом обозначений на рисунке определяется по формуле

$$k_\Phi = (\text{внутренний формат}) / (\text{внешний формат}) = \Phi_{\text{я}} / \Phi_{\text{ПУ}}.$$

Для буфера на рис. 11.10  $k_\Phi = 4$ . В общем случае  $k_\Phi$  может принимать значения, большее, меньшее или равное 1.



На рис. 11.11 представлена структура буфера с  $k_\Phi = 4$ , принимающего поток байт, поступающий из ПУ, например с дискового или ленточного ЗУ, и передающего 4-байтные слова в ОП. Рисунок позволяет более подробно представить функционирование буферного регистра  $P_2СИ$  в каналах ввода-вывода.

Каждый поступающий с шины интерфейса в буфер байт сопровождается сигналом, подаваемым на вход счетчика (по модулю 4) числа байт  $СчБ$ , который, переходя из одного состояния в другое, последовательно открывает группы вентилях, пропускающих байты в соответствующие позиции  $P_2СИ$ . При поступлении четвертого байта в  $P_2СИ$  завершается формирование 4-байтного слова,  $СчБ$  переходит в нулевое состояние и одновременно формируется сигнал запроса на передачу слова в ОП.

Благодаря наличию буфера в  $k_\Phi$  раз увеличивается интервал времени между запросами, или, другими словами, в  $k_\Phi$  раз уменьшается частота запросов на прием информации в ОП. Однако при этом не меняется максимально допустимое время  $\tau_{ПУ} \approx T_{ПУ}$  ожидания приема информации в ОП, так как уже через  $T_{ПУ}$  после формирования запроса к ОП в  $P_2СИ$  поступит байт 0 следующего слова и, если к этому моменту предыдущее слово не будет принято в ОП, произойдет искажение информации.

Можно говорить, что буфер виртуально изменяет количественные характеристики ПУ по отношению к устройствам ядра ЭВМ. В рассматриваемом на рис. 11.11 случае приведенные к ядру ЭВМ характеристики ПУ принимают следующие значения:

$$\Phi'_{ПУ} = 4\Phi_{ПУ}; T'_{ПУ} = 4T_{ПУ}; \Theta'_{ПУ} \approx 4T_{ПУ}; \tau'_{ПУ} \approx T_{ПУ}.$$

Если увеличить емкость буфера путем добавления дополнительного регистра  $P_2Д$  размером в 4-байтное слово (показан штриховой линией на рис. 11.11), как это сделано в структурной схеме селекторного канала (см. рис. 11.9), и после приема в  $P_2СИ$  четвертого байта производить передачу слова в  $P_2Д$ , одновременно формируя запрос к ОП, то максимальное время ожидания снятия информации из  $P_2Д$  возрастет до  $4T_{ПУ}$ . Таким образом, в результате увеличения емкости буфера (добавление  $P_2Д$ ) приведенные к ядру ЭВМ характеристики ПУ принимают следующие значения:

$$\Phi'_{ПУ} = 4\Phi_{ПУ}; T'_{ПУ} = 4T_{ПУ}; \Theta'_{ПУ} = 4T_{ПУ}; \tau'_{ПУ} = 4T_{ПУ}.$$

Увеличение  $\tau'_{ПУ}$  существенно для селекторных (блок-мультиплексных) каналов с интенсивным обменом информацией с ОП, особенно при реализации цепочек операций и данных.

Наличие буфера может приводить и к виртуальному качественному изменению характеристики ПУ. Например, в ЗУ с магнитной лентой передача при записи и считывании информации производится с постоянной скоростью; другими словами, ЗУ с магнитной лентой является устройством с синхронной передачей, навязывающей ОП свой темп приема (выдачи) информации. Однако, если есть буфер, емкость которого достаточно для промежуточного хранения блока (зоны) данных, считываемых (или записываемых) с ленты, то по отношению к ядру ЭВМ ЗУ с лентой становится устройством с асинхронной блочной передачей информации, работающим в произвольном темпе с остановкой ленты после передачи каждого блока.

## 11.7. Элементы организации интерфейсов

*Методы передачи информации между устройствами ЭВМ [79].* Используются два метода передачи дискретных сигналов: синхронный и асинхронный.

При синхронном методе (рис. 11.12) передающее устройство  $Y_1$  устанавливает одно из двух возможных состояний сигнала (0 или 1) и поддерживает его в течение определенного заранее выбранного времени, по истечении которого состояние сигнала на передающей стороне может быть изменено.

Время передачи сигнала, которое складывается из времени распространения сигнала по линии  $L_0$  и времени распознавания и фиксации сигнала в регистре приемного устройства  $Y_2$ , зависит от параметров линии связи и характеристик приемного и передающего устройств. Если обозначить через  $T$  максимальное время передачи сигнала (с учетом возможных наихудших условий), то для периода синхронной передачи информации должно выполняться условие

$$\tau \geq T. \quad (11.1)$$

При асинхронной передаче (рис. 11.13) устройство  $Y_1$  устанавливает

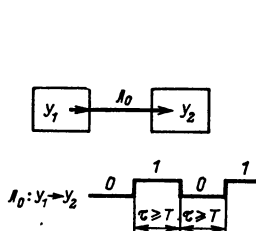


Рис. 11.12. Синхронный метод передачи информации

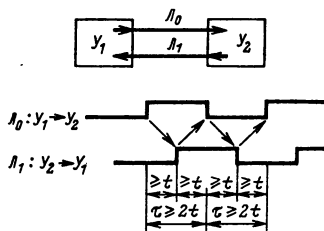


Рис. 11.13. Асинхронный метод передачи информации

соответствующее передаваемому коду состояние сигнала на линии  $L_0$ , а устройство  $U_2$  после приема нового состояния сигнала информирует об этом устройство  $U_1$  изменением состояния сигнала на линии  $L_1$ . Передающее устройство, получив сигнал о приеме, снимает передаваемый сигнал. Таким образом, период  $\tau$ , в течение которого передающее устройство должно поддерживать состояние сигнала, является переменным и зависит от характеристик конкретной линии связи и устройств, участвующих в передаче.

Если обозначить через  $t$  время передачи нового состояния сигнала в один конец линии связи, то при асинхронной передаче должно удовлетворяться условие

$$\tau \geq 2t. \quad (11.2)$$

Обычно время  $2t$  значительно меньше времени  $T$ , которое приходится выбирать, исходя из максимально возможных расстояний между устройствами.

При передаче параллельного кода по параллельным линиям сигналы поступят в приемное устройство в разное время из-за разброса параметров цепей, формирующих сигналы, и линий интерфейса. Максимальный разброс времени передачи

$$\Delta T = \max \{ |t_i - t_j| \}. \quad (11.3)$$

Рассмотрим два метода передачи параллельного кода по нескольким линиям: *со стробированием*, при котором используется синхронная передача, и *с квитиowaniem*, при котором используется асинхронная передача.

На рис. 11.14 приведена временная диаграмма для передачи со стробированием. Информация передается по линиям  $L_1, \dots, L_n$  в интервале времени, когда сигнал на линии  $L_0$  соответствует 1. При нулевом сигнале на линии  $L_0$  сигналы на шинах  $L_1, \dots, L_n$  не имеют смысла.

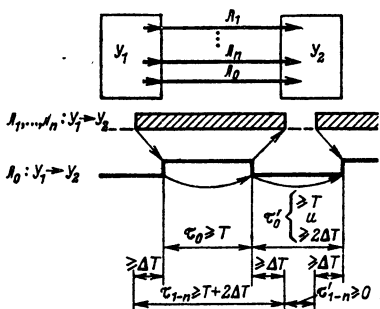


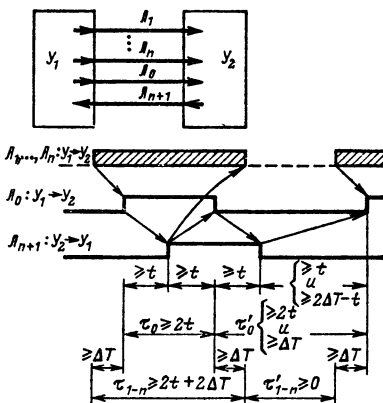
Рис. 11.14. Передача информации со стробированием

Для гарантированной передачи данных по линиям  $L_1, \dots, L_n$  передаваемый код устанавливается как минимум на время  $\Delta T$  раньше, чем появляется единичный сигнал на линии  $L_0$ . Снятие передаваемого кода с линий  $L_1, \dots, L_n$  может закончиться на время  $\Delta T$  позже времени установления нулевого состояния на линии  $L_0$ . На рисунке приведены необходимые временные соотношения между сигналами.

Передача параллельного кода с квитиowaniem показана на рис. 11.15.

Рис. 11.15. Передача информации с квитируанием

При передаче с квитируанием фронт сигнала на линии  $L_0$  сигнализирует устройству  $U_2$ , что устройство  $U_1$  подготовило передаваемую информацию на линиях  $L_1, \dots, L_n$ . Фронт сигнала на линии  $L_{n+1}$  сигнализирует устройству  $U_1$ , что устройство  $U_2$  приняло передаваемую информацию. Восприняв фронт сигнала по линии  $L_{n+1}$ , устройство  $U_1$  снимет информацию с линий  $L_1, \dots, L_n$  и погасит сигнал на линии  $L_0$ , срез которого сигнализирует устройству  $U_2$  об окончании передачи данных. После снятия сигнала на линии  $L_0$  устройство  $U_2$  погасит сигнал на линии  $L_{n+1}$ , срез которого сигнализирует устройству  $U_1$  о готовности устройства  $U_2$  к приему следующей порции данных.



Передача с квитируанием обычно используется, когда приемное устройство  $U_2$  не всегда готово к приему информации (занято выполнением других операций). Этот способ передачи применяется в интерфейсе ввода-вывода ЕС ЭВМ и др.

Передача со стробированием используется главным образом для пересылок информации внутри устройства, например между регистрами. В этом случае еще в процессе проектирования устройства для передачи информации можно выбрать такие моменты времени, когда участвующие в передаче элементы свободны и готовы к этой операции.

*Структура шин и линии интерфейса* [79]. Выше рассмотрены методы передачи информации между двумя устройствами. Однако при проектировании ЭВМ приходится решать более сложную задачу — организацию передачи информации в группе взаимосвязанных устройств. Характерным является случай централизованной связи, когда передача информации производится только между устройством  $U_0$  и одним из устройств  $U_1, \dots, U_n$ . Примером является передача информации между каналом и ПУ.

При организации связи группы устройств возникает необходимость в адресации и идентификации устройств  $U_1, \dots, U_n$ . *Адресация* в данном случае состоит в выборе центральным устройством  $U_0$  одного из устройств  $U_1, \dots, U_n$  для связи. *Идентификация* состоит в определении центральным устройством, какое из устройств  $U_1, \dots, U_n$  запрашивает связь.

Адресация и идентификация устройства осуществляются путем передачи соответствующей информации по линиям интерфейса.

Различные структуры линий и шин интерфейса можно классифицировать: индивидуальные, коллективные, комбинированные.

Наиболее надежной является структура с индивидуальными линия-

ми и шинами, поскольку выход из строя одной группы линий и шин не влияет на работу других устройств. При использовании индивидуальных линий и шин упрощаются адресация и идентификация, но увеличивается количество оборудования. Индивидуальные линии и шины используются в основном для связи вычислительной машины с устройствами технологической автоматики.

Структура с коллективными шинами и линиями имеет меньшую надежность, но при необходимости организации связи с большим числом устройств такое выполнение позволяет уменьшить объем оборудования.

Применение коллективных линий и шин интерфейса возможно в том случае, если передача между различными устройствами происходит по единым установленным правилам. В интерфейсе ввода-вывода современных ЭВМ главным образом используется система коллективных линий и шин. В некоторых случаях используется комбинированная система индивидуальных и коллективных линий и шин.

На рис. 11.16 представлена структура с индивидуальными линиями и шинами. Жирными линиями изображены шины, по которым передаются данные.

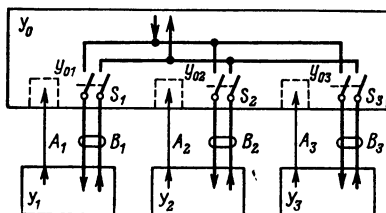
Центральное устройство  $U_0$  с любым устройством  $U_i$  связывается с помощью индивидуальных линий  $A_i$  и шин  $B_i$ .

Устройство  $U_0$  имеет переключатели  $S_i$  для подключения шин  $B_i$ . На рисунке переключатели изображены в виде электромеханических контактов, однако такие переключатели реализуются в виде электронных устройств.

Для адресации  $U_i$  устройство  $U_0$  должно включить соответствующий переключатель  $S_i$ .

Идентификация устройства  $U_i$  осуществляется следующим образом: сначала  $U_i$  на линии  $A_i$  возбуждает сигнал требования на установление связи, затем соответствующий узел  $U_{0i}$  устройства  $U_0$  определяет, от какого устройства пришел сигнал требования. Как только устройство  $U_0$  готово к обмену информацией, замыкается переключатель  $S_i$  и начинается передача данных. Передача информации производится одним из методов, рассмотренных выше.

На рис. 11.17 представлена структура с коллективными линиями и шинами. По коллективной шине  $B$  происходит обмен информацией между  $U_0$  и  $U_i$ , по коллективной линии  $A$  из  $U_i$  в  $U_0$  передается сигнал



требования на установление связи. Кроме того, имеется коллективная линия  $D$ , которая выходит из  $U_0$ , последовательно проходит через устройства  $U_i$  и возвращается в устройство  $U_0$ . При адресации  $U_i$  устройство

Рис. 11.16. Структура с индивидуальными линиями и шинами

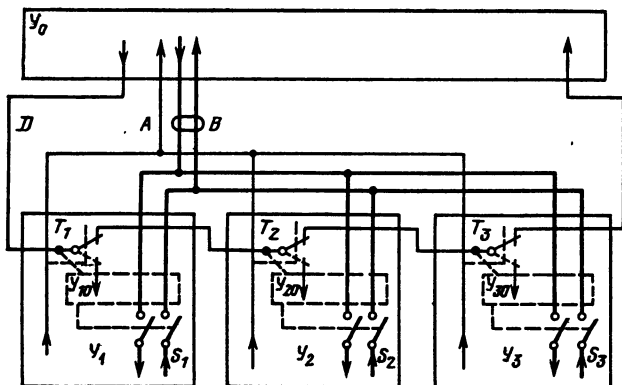


Рис. 11.17. Структура с коллективными линиями и шинами

$Y_0$  устанавливает на шинах  $B$  код номера устройства  $Y_i$  и посылает сигнал «выборки» по линии  $D$ . Если код номера на шинах  $B$  не совпадает с номером устройства  $Y_i$ , то переключатель  $T_i$  остается в исходном состоянии и сигнал по линии  $D$  распространяется на следующее устройство:  $Y_{i+1}$ . При совпадении кода с номером устройства переключатель  $T_i$  замыкается, дальнейшее распространение сигнала по линии  $D$  прекращается, а выбранное устройство  $Y_i$  соединяется с  $Y_0$  путем замыкания переключателя  $S_i$ . Если сигнал, посылаемый по линии  $D$ , возвращается в  $Y_0$ , то это означает, что адресованное устройство  $Y_i$  не найдено (обычно это свидетельствует о неисправности в работе интерфейса).

Необходимость в идентификации возникает, если устройство  $Y_i$  по коллективной линии  $A$  передает сигнал запроса связи.

Когда устройство  $Y_0$  готово к обслуживанию  $Y_i$ , оно посылает сигнал выборки по линии  $D$ . Устройства, не посылавшие запрос, пропускают сигнал по линии  $D$ . Первое на пути сигнала выборки устройство  $Y_i$ , пославшее запрос, замыкает переключатели  $T_i$  и  $S_i$  и по шинам  $B$  посылает в  $Y_0$  код собственного номера.

Приведенная на рис. 11.17 структура коллективных линий и шин подобна используемой в интерфейсе ввода-вывода ЕС ЭВМ. Другие варианты структуры коллективных линий и шин будут приведены при рассмотрении интерфейса малых и микроЭВМ.

## 11.8. Интерфейс ввода-вывода ЕС ЭВМ

В интерфейсе ввода-вывода ЕС ЭВМ применяется система коллективных линий и шин. Передача информации между устройствами производится с квити́рованием. Сигналы на линиях и шинах интерфейса электрически доступны всем ПУ, однако

логика работы интерфейса строится таким образом, что в каждый момент времени только одно из ПУ может быть логически связано с интерфейсом и реагировать на сигналы интерфейса. Если два или более ПУ требуют обслуживания, то из них выбирается одно в соответствии с установленным между ними приоритетом.

Периферийные устройства подсоединяются к интерфейсу через УПУ, которые могут быть групповыми (разделенными), обслуживающими несколько ПУ, или индивидуальными, обслуживающими одно ПУ. Применяют также двухканальные и многоканальные групповые УПУ, способные производить ввод и вывод информации через два или более канала, принадлежащих одной или несколькими ЭВМ.

Несколько ПУ могут через коммутатор присоединяться к нескольким групповым УПУ, при этом каждое ПУ может работать с любым свободным в данный момент УПУ. Своеобразным ПУ для канала и интерфейса является адаптер «канал — канал», через который осуществляется связь между каналами одной или разных ЭВМ.

*Информация, передаваемая через интерфейс.* Через интерфейс информация передается параллельным кодом побайтно, включая контрольный разряд. В *расширенном интерфейсе* (см. § 11.9) возможна двухбайтная передача. От канала через интерфейс в ПУ (точнее, в УПУ) передаются выводимые байты данных, приказы и номер (адрес) ПУ. От УПУ в канал поступают вводимые байты данных, номер (адрес) ПУ, коды состояния (*байт состояния и байты уточненного состояния*).

Коды состояния служат для опознавания процессором и каналом (и операционной системой) текущего, в том числе ненормального состояния ПУ. Байт состояния содержит основную информацию о состоянии данного ПУ. Значения отдельных разрядов (указателей) байта состояния приведены в табл. 11.1. Указатели формируются ПУ или УПУ.

Указатель «ПУ кончило» устанавливается при завершении операции ввода-вывода ПУ, а также при переходе ПУ из состояния «Не готов»

Т а б л и ц а 11.1

Номер разряда	Указатель	Номер разряда	Указатель
К	Контрольный разряд	4	Канал кончил
0	Внимание	5	ПУ кончило
1	Модификатор	6	Сбой в устройстве
2	УПУ кончило	7	Особый случай
3	Занято		

в состояние «Готов». Периферийное устройство находится в состоянии «Не готов», если необходимо вмешательство оператора (нет перфокарт, бумаги, не заправлена лента в ПУ и т. п.). Указатель «УПУ кончило» формируется (для групповых УПУ), если при обращении к нему оно было занято.

Указатель «Канал кончил» устанавливается при завершении передачи через интерфейс данных текущей операции ввода-вывода. Указатель «Модификатор» формируется в том числе при заданном в УПУ условии окончания операции ввода-вывода (например, при поиске по ключу в ВЗУ на дисках).

Указатель «Сбой в устройстве» устанавливается при ошибках в ПУ и УПУ. Характер ошибки уточняется байтами уточненного состояния. Указатель «Особый случай» имеет определенное значение для каждого приказа и типа ПУ и служит для указания некоторого ненормального положения в ПУ.

Условия формирования указателя «Внимание» зависят от типа ПУ. При установке этого указателя прерывается выполнение цепочки УСК.

Байты уточненного состояния, содержащие информацию об ошибке во время выполнения последней операции ввода-вывода, передаются из ПУ в канал по приказу «Уточнить состояние». Число передаваемых при этом байт и значение информации зависят от типа ПУ.

*Нумерация (адресация) ПУ.* Канал различает ПУ по присвоенным

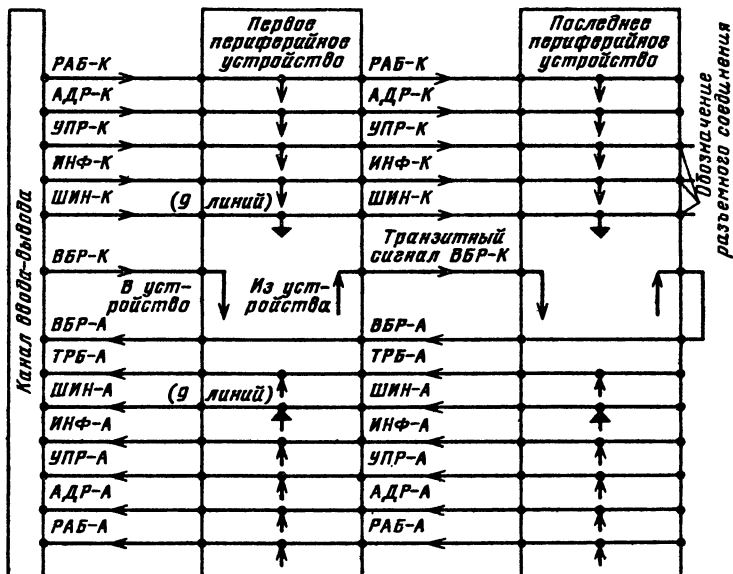


Рис. 11.18. Основные шины и линии интерфейса ввода-вывода



номерам. Нумерация ПУ производится 8-разрядными номерами, фиксируемых с помощью встроенных в ПУ тумблерных регистров или специальных коммутационных плат. Следует иметь в виду, что нумерация ПУ обычно не определяет приоритет между ними в использовании интерфейса. Приоритет задается порядком присоединения ПУ к интерфейсу.

*Основные шины и линии интерфейса* представлены на рис. 11.18. Для обозначения ПУ, связанного с каналом через интерфейс, используется термин «абонент»; соответственно сигналы, поступающие в канал из интерфейса, именуют сигналами абонента.

В интерфейсе выделяют две группы информационных шин:

1) шина прямой передачи данных (ШИН-К), по которой из канала в ПУ передаются выводимый слог (байт) информации, приказ и номер (адрес) ПУ;

2) шина обратной передачи (ШИН-А), по которой из ПУ в канал передаются вводимый слог (байт) информации, байты состояния и номер (адрес) ПУ.

Информация, передаваемая по информационным шинам, контролируется по четности. Для этого в каждой шине отводится дополнительная линия для передачи контрольного разряда (см. гл. 12).

Смысл информации на шинах прямой и обратной передач и промежутков времени, в течение которого информация на шинах имеет тот или иной смысл, определяются служебными сигналами, называемыми сигналами идентификации.

К каждой информационной шине относятся три сигнала идентификации — «Адрес», «Управление» и «Информация» — и соответствующие им линии. Сигналы идентификации от канала, относящиеся к шине прямой передачи, обозначим АДР-К, УПР-К и ИНФ-К, сигналы идентификации от абонента, относящиеся к шине обратной передачи, — АДР-А, УПР-А и ИНФ-А.

Сигналы АДР-К, УПР-К и ИНФ-К используются для идентификации появления на шине прямой передачи соответственно номера ПУ, приказа и байта данных; сигналы АДР-А, УПР-А и ИНФ-А служат для идентификации появления на шине обратной передачи соответственно номера ПУ, байта состояния и байта данных.

Кроме указанных в интерфейсе используются также следующие линии и сигналы управления:

линия и сигнал «Работа канала» — РАБ-К. Сигнал РАБ-К передается от канала к подсоединенным ПУ и используется для разрешения подключения ПУ к каналу. Сброс сигнала РАБ-К вызывает сброс всех сигналов абонента и прекращение всех операций, выполняемых в это время через интерфейс;

линия и сигнал «Работа абонента» — РАБ-А. Линия соединяет все ПУ с каналом. Сигнал РАБ-А указывает, что ПУ подключилось к нему. Сигнал РАБ-А должен сохраняться, пока требуемая в сеансе

связи передача информации между каналом и ПУ не будет завершена;

линия и сигнал «Требование абонента» — ТРБ-А. Линия ТРБ-А соединяет все ПУ с каналом. Сигнал ТРБ-А сигнализирует о том, что ПУ требуется установить связь с каналом. Сигналы требования могут возбуждать несколько ПУ в одно и то же время. Обычно сигнал ТРБ-А, выданный данным ПУ, сбрасывается после выдачи этим ПУ сигнала РАБ-А;

линия и сигналы «Выборка канала» — ВБР-К и «Выборка абонента» — ВБР-А. Линии ВБР-К и ВБР-А образуют замкнутую цепь, которая выходит из канала (линия ВБР-К), последовательно проходит через все ПУ, начиная с устройства с высшим и кончая устройством с низким приоритетом, после чего возвращается в канал в виде линии ВБР-А.

Все сигналы, поступающие на шины и линии интерфейса и принимаемые с интерфейса, усиливаются и формируются усилителями-передатчиками и усилителями-приемниками.

Каждое ПУ может выдать сигнал РАБ-А только при наличии на его входе сигнала ВБР-К. Если данное ПУ не участвует в выборке, оно пропускает сигнал ВБР-К на следующее устройство, после чего данное ПУ не может выдать сигнал РАБ-А до следующего поступления сигнала ВБР-К. В ответ на снятие сигнала ВБР-К снимается сигнал РАБ-А.

Сигнал ВБР-К является единственным сигналом, который недоступен сразу всем абонентам. Поступление в канал сигнала ВБР-А во время выполнения процедуры начальной выборки ПУ свидетельствует об отсутствии или неисправности адресуемого ПУ.

Набор шин интерфейса, представленных на рис. 11.18, является упрощенной интерпретацией интерфейсов реальных ЭВМ. В действитель-

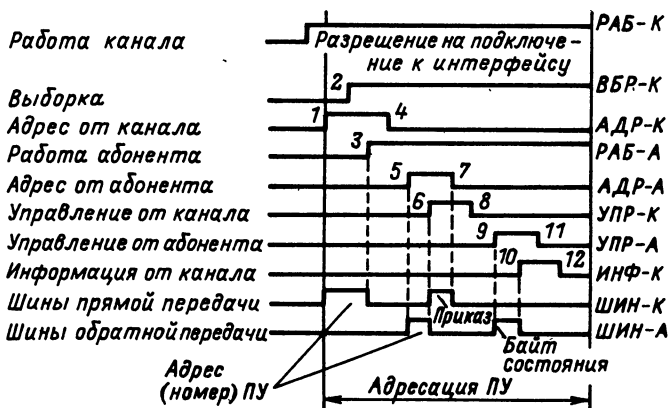


Рис. 11.19. Временная диаграмма процедуры начальной выборки периферийного устройства при пуске операции ввода-вывода в селекторном канале

ности обычно имеются дополнительные линии и сигналы управления (разрешение выборки и блокировки), а также линии и сигналы для управления измерением времени и сменой состояния ПУ (измерение от канала и абонента, смена состояния).

*Основные процедуры интерфейса.* Основными процедурами являются начальная выборка ПУ и обслуживание ПУ. Рассмотрим в качестве примера протокол процедуры начальной выборки ПУ для пуска операции ввода-вывода в селекторном канале (рис. 11.19).

1. Канал выдает на шину прямой передачи адрес (номер) ПУ и возбуждает сигнал АДР-К. Каждое ПУ, подсоединенное к каналу, дешифрует номер, но только одно устройство опознает его как свой номер.

2. Канал выдает сигнал ВБР-К, который проходит последовательно по ПУ до тех пор, пока не достигнет адресуемого устройства. В случае его отсутствия или неисправности в канал поступает сигнал ВБР-А.

3. Периферийное устройство, обнаружившее совпадение номеров, блокирует дальнейшее распространение сигнала ВБР-К и выдает в канал сигнал РАБ-А.

4. После прихода в канал сигнала РАБ-А канал сбрасывает сигнал АДР-К.

5. После снятия сигнала АДР-К ПУ выдает на шины обратной передачи свой номер и сигнал идентификации АДР-А.

6. Получив сигнал АДР-А, канал проверяет полученный им номер ПУ и при совпадении этого номера с заданным выдает на шину прямой передачи байт, представляющий собой приказ для ПУ, после чего возбуждает сигнал идентификации УПР-К.

7. Выбранное ПУ принимает приказ в свой регистр и сбрасывает сигнал АДР-А.

8. В ответ на сброс сигнала АДР-А канал сбрасывает сигнал УПР-К.

9. После снятия сигнала УПР-К ПУ выдает на шину обратной передачи байт состояния и формирует сигнал идентификации УПР-А.

10. Получив сигнал УПР-А, канал анализирует байт состояния ПУ. Если байт содержит нули во всех разрядах (кроме контрольного), что указывает на готовность ПУ, канал отвечает сигналом ИНФ-К.

11. В ответ на выдачу сигнала ИНФ-К ПУ сбрасывает сигнал УПР-А.

12. После снятия сигнала УПР-А канал сбрасывает сигнал ИНФ-К, завершая этим последовательность сигналов начальной выборки ПУ при пуске операции в селекторном режиме.

При работе канала в мультиплексном режиме после начальной выборки сеанс связи с ПУ заканчивается (канал снимает сигнал ВБР-К, а устройство — сигнал РАБ-А). Канал освобождается и может начинать процедуру начальной выборки или обслуживания других устройств. В селекторном канале после завершения начальной выборки начинается передача данных между ПУ и каналом.

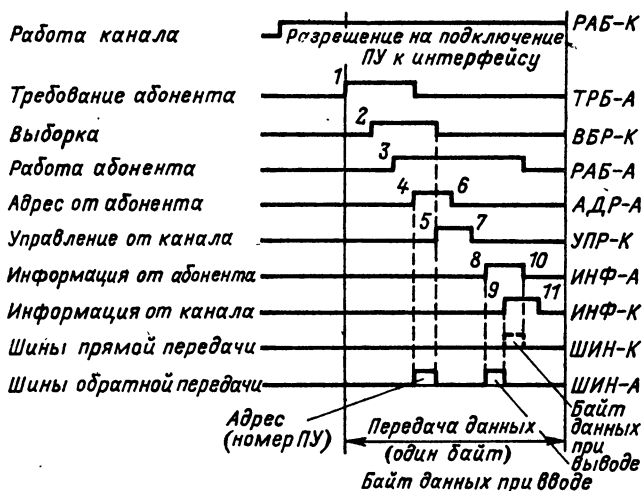


Рис. 11.20. Временная диаграмма процедуры обслуживания при вводе-выводе информации через байт-мультиплексный канал

Рассмотрим протокол процедуры обслуживания ПУ при вводе (выводе) данных через байт-мультиплексный канал (рис. 11.20).

1. Периферийное устройство, после того как оно готово к передаче информации, возбуждает сигнал ТРБ-А.

2. Канал, после того как сеанс по требованию ПУ окажется возможным, посылает сигнал ВБР-К (в этом режиме работы канал осуществляет выборку ПУ без выдачи номера ПУ на шину прямой передачи и без формирования сигнала идентификации АДР-К).

Появление сигнала ВБР-К без предварительной установки сигнала АДР-К является для ПУ признаком, отличающим процедуру начальной выборки (адресации) от процедуры обслуживания (идентификации).

3. Сигнал ВБР-К проходит последовательно по всем ПУ до тех пор, пока не достигнет первого устройства, пославшего сигнал ТРБ-А и ждущего сеанса связи. На этом распространение сигнала ВБР-К прекращается. Выбранное ПУ выдает сигнал РАБ-А.

4. Периферийное устройство выдает на шину обратной передачи свой номер и возбуждает сигнал идентификации АДР-А.

Этапы работы интерфейса 5—7 в данной процедуре такие же, как и этапы 6—8 в временной диаграмме на рис. 11.19. Отличие состоит в том, что формирование каналом сигнала УПР-К означает не выдачу байта приказа ПУ на шину прямой передачи, а указание ПУ продолжать работу, связанную с текущей операцией ввода-вывода. Кроме того, на этапе 5 канал после получения сигнала АДР-А снимает сигнал ВБР-К; на этапе 6 проверка номеров отсутствует, а вместо этого из памяти под-

каналов выбирается управляющая информация о текущих параметрах подканала. Этапы работы интерфейса 8—11 рассматриваемой временной диаграммы соответствуют этапам 9—12 на временной диаграмме на рис. 11.19. Отличие состоит в том, что на шину обратной передачи выдается не байт состояния, а байт информации и вместо сигнала идентификации УПР-А формируется сигнал ИНФ-А. Кроме того, на этапе 10 наряду со снятием сигнала ИНФ-А гасится сигнал РАБ-А. На этом последовательность передачи данных через интерфейс в мультиплексном режиме завершится.

Временная диаграмма обслуживания ПУ при выводе данных отличается от рассмотренной только тем, что байт данных передается по шине прямой передачи, причем эта передача производится в интервале времени между фронтом сигнала ИНФ-К и срезом сигнала ИНФ-А (показано штриховыми линиями на рис. 11.20).

Отметим, что, как следует из рассмотренных диаграмм, некоторые сигналы идентификации на ряде этапов работы интерфейса используются не по прямому назначению, а в качестве квитирующих сигналов.

## 11.9. Развитие системы ввода-вывода в ЕС ЭВМ

Повышение производительности моделей машин в ЕС ЭВМ и применение ЗУ прямого доступа на дисках со скоростью передачи информации около 1 Мбайт/с и более потребовали дальнейшего развития системы ввода-вывода в направлении повышения пропускной способности интерфейса ввода-вывода, повышения уровня параллелизма в работе внешних ЗУ, придания системе ввода-вывода новых логических возможностей.

В систему ввода-вывода ЕС ЭВМ внесен ряд архитектурных новшеств, среди которых отметим следующие:

- мультиплексирование блоков данных при работе с внешними ЗУ;
- расширенный интерфейс ввода-вывода;
- дополнительные команды ввода-вывода;
- повторные выполнения управляющего слова канала;
- селективный сброс, вводимый УПУ;
- косвенная адресация данных в канале.

*Мультиплексирование блоков данных* при операциях ввода-вывода с внешними ЗУ реализуется блок-мультиплексными каналами (см. § 11.5), в которых создаются несколько подканалов. Режим мультиплексирования возможен, если разряд «управление мультиплексированием» управляющего регистра 0 установлен в 1. В противном случае канал работает как селекторный. На рис. 11.21 показан процесс мультиплексирования блоков данных при работе блок-мультиплексного канала с двумя периферийными устройствами: ПУ1 и ПУ2 [49], при этом для каждого ПУ выполняется своя канальная программа. Следует под-

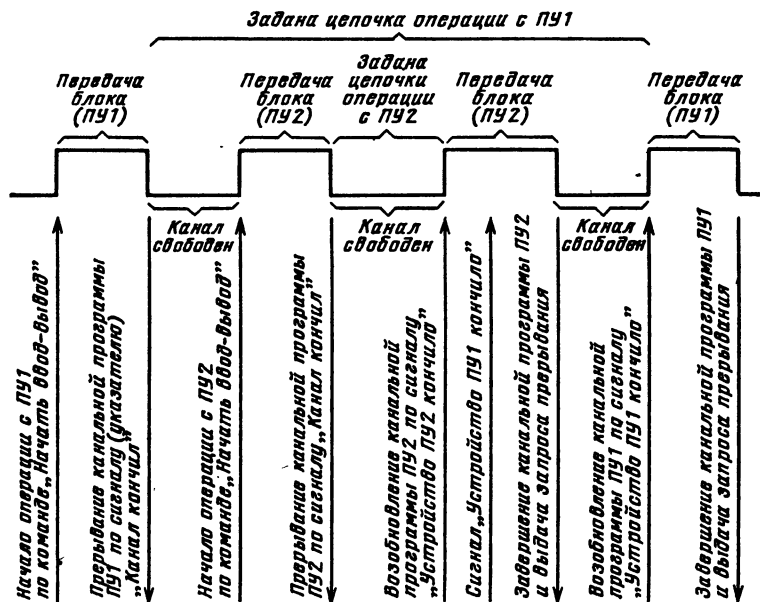


Рис. 11.21. Мультиплексирование блоков данных в блок-мультиплексном канале ввода-вывода

черкнуть, что сама передача блока данных производится в монопольном режиме.

*Расширенный интерфейс ввода-вывода.* В целях повышения пропускной способности интерфейса в некоторых блок-мультиплексных каналах может применяться параллельная передача двухбайтных кодов и повышенная частота передачи информации в интерфейсе.

Это достигнуто увеличением числа информационных шин интерфейса для передачи двухбайтных кодов и введением новых линий и сигналов управления и идентификации, позволяющих увеличить частоту передачи данных через интерфейс до 3 млн.байт/с, при этом сохраняется возможность работы с ПУ с однобайтной передачей данных через интерфейс.

Помимо имеющихся в однобайтном интерфейсе шин из девяти линий для однобайтной передачи ШИН-К и ШИН-А (именуемых в расширенном интерфейсе ШИН-КО и ШИН-АО) добавлены шины из девяти линий ШИН-К1 и ШИН-А1 для передачи параллельно вторых байт.

Введены линии и сигналы маркеров МРК-КО, МРК-АО, МРК-К1, МРК-А1, указывающих, на каких комплектах шин установлены передаваемые байты.

Введены также дополнительные линии и сигналы идентификации ДАН-А и ДАН-К, аналогичные линиям и сигналам ИНФ-А и ИНФ-К. Сигналы ДАН-А и ДАН-К посылаются в моменты,

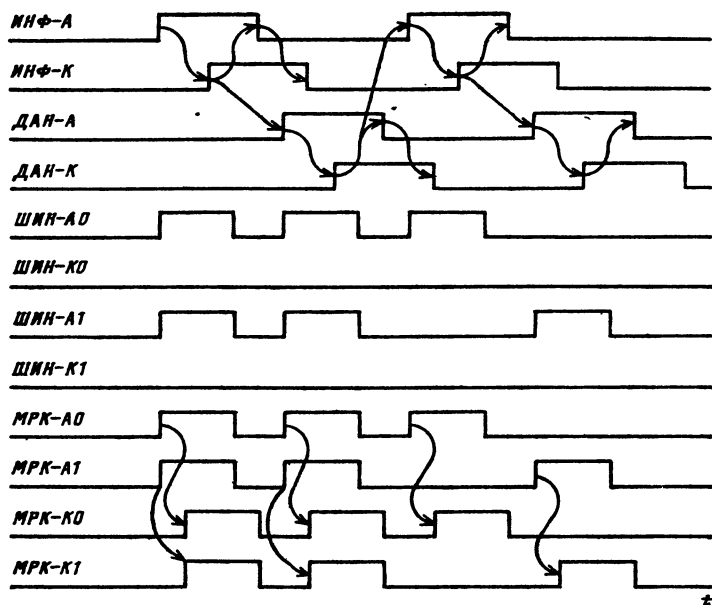


Рис. 11.22. Временная диаграмма процедуры считывания в расширенном интерфейсе ввода-вывода (ЕС ЭВМ)

когда сбрасываются сигналы **ИНФ-А** и **ИНФ-К**, а сигналы **ИНФ-А** и **ИНФ-К** устанавливаются, когда сбрасываются сигналы **ДАН-А** и **ДАН-К** и за этот счет увеличивается частота передачи данных по линиям интерфейса. Дополнительные сигналы **ДАН-А** и **ДАН-К** могут использоваться для увеличения скорости (частоты) передачи данных блок-мультиплексными каналами, работающими с однобайтным интерфейсом.

На рис. 11.22 приведена временная диаграмма процедуры считывания (ввода) в расширенном интерфейсе ввода-вывода ЕС ЭВМ.

Дополнительные команды ввода-вывода выполняются в привилегированном режиме, если разряд «управление мультиплексированием» управляющего регистра 0 установлен в 1. К дополнительным относятся следующие команды ввода-вывода.

«Начать ввод-вывод с быстрым отключением». Процессор не ждет окончания процедуры начальной выборки, как это бывает при выполнении обычной команды «Начать ввод-вывод», а сразу после выборки каналом АСК освобождается и может продолжить выполнение программы.

**«Освободить ввод-вывод».** Операция в указанном в команде ПУ прекращается, состояние операции в этот момент отображается в ССК, и подканал переходит в состояние «доступен». Команда быстрее переводит подканал в состояние «доступен», чем обычная команда «остановить ввод-вывод», после выполнения которой для освобождения подканала процессор должен обработать прерывание от соответствующего подканала.

**«Остановить устройство»** — команда прекращает операцию только в адресованном командой ПУ, а сам канал продолжает работу с другими ПУ.

**«Записать идентификатор канала».** Команда записывает в выделенные фиксированные ячейки ОП сведения о типе и состоянии адресуемого канала.

*Повторение управляющего слова канала* ускоряет повторение ошибочно выполняемой операции. При ошибке в операции ПУ может запросить повторение УСК, причем для повторения не требуется обработка запроса прерывания ввода-вывода. Периферийное устройство запрашивает повторение УСК выдачей соответствующего байта состояния и одновременно сигнала на линии МРК-А0.

*Селективный сброс*, вводимый блоком управления ПУ, позволяет прервать «зависание интерфейса», вызываемое нарушением из-за сбоев или отказов в ПУ или канале нормальной последовательности сигналов, которыми должны обмениваться канал и ПУ при операции ввода-вывода. Селективный сброс производится по запросу УПУ, выставяющего сигнал на специальной линии «отключение от абонента», в ответ на который канал выполняет «селективный сброс».

*Косвенная адресация данных в канале* дополняет процессорные средства реализации виртуальной памяти (динамического преобразования адресов) канальными средствами динамического преобразования адресов данных при операциях ввода-вывода. Благодаря этим средствам одно УСК может осуществлять обмен данными с несмежными страницами физической памяти.

Режим косвенной адресации (КАД) имеет место, если УСК [37] = 1. В режиме КАД поле текущего адреса данных в УСК (УСК [8—31]) указывает не адрес данных в ОП, а адрес в ОП первого слова в списке *слов косвенной адресации данных*. В поле текущего счетчика данных указывается число принадлежащих данному УСК слов косвенной адресации, которые размещаются в последовательных ячейках ОП. Каждое слово КАД (кроме первого) указывает физический адрес начала блока памяти (2048 байт), а первое слово — физический адрес внутри блока.

Как только при операции ввода-вывода обнуляются 11 младших разрядов физического адреса ячейки, участвующей в обмене, выбирается следующее слово КАД. Оно задает начальный адрес нового блока ОП, который следует использовать в операции обмена.



## 11.10. Структура и временные диаграммы интерфейса «Q-шина» малых ЭВМ

К интерфейсам малых и микроЭВМ, в основном предназначенных для работы в системах реального времени, предъявляются повышенные требования в отношении простоты, гибкости и высокой динамичности. Для этих машин характерным решением является общий интерфейс, при котором один и тот же набор линий обеспечивает связь между процессором, основной памятью и периферийными устройствами. Для всех передач информации между любыми устройствами используют одни и те же линии, процедуры, команды, управляющие сигналы. Передача информации между устройствами выполняется в режиме разделения во времени интерфейса.

В каждый момент времени передача информации через интерфейс может происходить только между двумя устройствами, причем одно из них является задатчиком (ведущим), а другое — исполнителем. Исполнителем может быть любое устройство, подсоединенное к интерфейсу, а задатчиком — любое, кроме модулей памяти. Задатчик получает интерфейс в свое распоряжение на время выполнения процедуры передачи данных.

Роли устройств в интерфейсе в процессе работы ЭВМ непрерывно меняются. Например, при выборе операнда из памяти процессор является задатчиком, а память — исполнителем. При получении сигнала прерывания процессор является исполнителем, а задатчиком — ПУ (источник прерывания).

Для обеспечения гибкости и высокой динамичности во многих интерфейсах подобного типа реализованы следующие свойства: наличие передач как программно-управляемых, так и внепроцессорных (прямой доступ к памяти);

адресуемость регистров ПУ (точнее, УПУ) подобно ячейкам ОП;

возможность присвоения устройствам ЭВМ нескольких уровней приоритета в занятии интерфейса, которые могут изменяться пересоединением линий запросов, а в процессоре — программным путем;

совмещение по времени процедуры выбора с учетом приоритетов следующего устройства для предоставления интерфейса (процедуры арбитража) с обслуживанием интерфейсом предыдущего;

единая нумерация ячеек памяти, адресуемых регистров процессора и регистров (приемников и источников информации) периферийных устройств, позволяющая в операциях ввода-вы-

вода обходиться без специальных команд и использовать обычные адресные команды и все способы адресации. Таким образом, данные в регистрах ПУ становятся непосредственно доступными процессору без предварительной передачи их в его регистры или в оперативную память. Имеется возможность прямой передачи данных из регистра одного ПУ в регистр другого.

Особенностью интерфейсов рассматриваемого типа является органическое включение в него средств систем прерывания (см. гл. 9).

К общим шинным интерфейсам, используемым в малых ЭВМ, относятся интерфейс «общая шина» (Unibus) и его модификация «Q-шина».

В предыдущем издании книги [22] приведен материал по организации интерфейса «общая шина» (Unibus фирмы DEC, США), широко использовавшегося в прошлые годы в малых ЭВМ (например, в малых ЭВМ СМ-4). В настоящее время машины с параметрами, еще недавно характерными для малых ЭВМ, выполняются в виде микроЭВМ, основу которых составляют микропроцессорные БИС. Ограничение на число выводов корпусов этих БИС заставило искать пути упрощения интерфейса «общая шина» и в первую очередь уменьшения числа линий в интерфейсе. В результате фирмой DEC была предложена упрощенная модификация интерфейса «общая шина», получившая название «Q-шина», первоначально использовавшаяся в простейших одноплатных микроЭВМ с одноуровневым прерыванием (LSI-11, «Электроника-60»). Впоследствии значительное развитие логических возможностей интерфейса «Q-шина» превратило его в типовой системный интерфейс для малых и микроЭВМ с архитектурой, подобной архитектуре машин фирмы DEC (см. гл. 9). При этом развитии учитывалась необходимость обеспечить совместимость новых модификаций «Q-шины» с ранее выпущенными вычислительными средствами.

Рассмотрим организацию интерфейса «Q-шина», который во многом подобен интерфейсу «общая шина»<sup>1</sup>. Однако если в интерфейсе «общая шина» используются 56 линий, то в интерфейсе «Q-шина» при более широких по ряду характеристик логических возможностях (например, больше адресное пространство шины) — всего 42 линии. Это достигнуто в первую очередь применением мультиплексируемой шины для передачи адреса и данных, а также уменьшением с четырех линий до одной линии сигналов разрешения прерывания (см. гл. 9).

---

<sup>1</sup> При описании интерфейса «Q-шина» использованы материалы, предоставленные автору инженером Н. В. Сосиной.

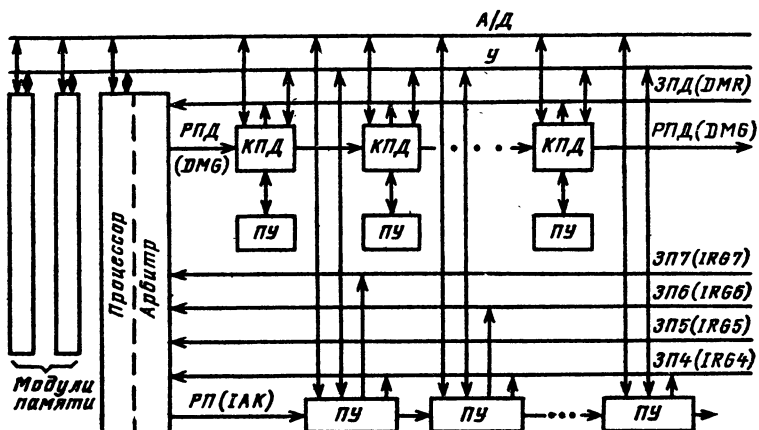


Рис. 11.23. Упрощенная структура интерфейса «Q-шина»:  
А/Д — линии передачи адреса и данных; У — линии управления

Упрощенная структура интерфейса «Q-шина» представлена на рис. 11.23.

42 линии «Q-шина» можно разбить на следующие группы (наименования линий и передаваемых по ним сигналов совпадают):

а) группа линий для передачи данных и адреса (А/Д), объединяющая:

мультиплексируемую шину данные/адрес, состоящую из 16 двунаправленных линий *DAL* [15 ÷ 00], по которым может производиться обмен 16-разрядными словами или байтами;

две мультиплексируемые двунаправленные линии четность памяти/адрес *DAL* [17 ÷ 16]. По этим линиям при считывании из памяти передается сигнал ошибки по четности, вызывающий внутреннее прерывание;

четыре дополнительные адресные линии *DAL* [21 ÷ 18];

б) группа линий управления (У), объединяющая:

шесть линий управления передачей адреса и данных: синхронизация *SYNC*, строб входной информации *DIN*, строб выходной информации *DOU*T, запись байта *WTBT*, выбор периферийного устройства *BS7*, ответ *RPLY*;

шесть линий системного управления: питание в норме *POK*, постоянное напряжение в норме *DCOK*, инициализация *INIT*, останов процессора *HALT*, регенерация памяти *REF*, *EVNT*;

в) группа линий управления прерываниями и прямым доступом к памяти, содержащая:

пять линий управления прерываниями: *линии запросов прерывания IRQ4, IRQ5, IRQ6, IRQ7 и линия разрешения прерывания IAK*;

три линии управления прямым доступом к памяти (ПДП): *запрос ПДП DMR, разрешения ПДП DMG, подтверждения разрешения ПДП SACK*.

Наличие дополнительных адресных линий позволяет в рассматриваемом интерфейсе использовать 16-, 18- и 22-разрядные адреса, т. е. расширять емкость памяти от 64 Кбайт до 4 Мбайт.

Путем введения сигнала *BS7*, указывающего на участие в передаче периферийного устройства, преодолен недостаток интерфейса «общая шина», связанный с фиксированным расположением «страницы ввода-вывода» (адресов регистров ПУ) в области адресов памяти 7700—7600, что затрудняло расширение адресного пространства памяти. Сигнал *BS7* указывает также, что адрес регистра ПУ задается старшими 12 разрядами адреса.

В интерфейсе реализуются следующие операции: «Запись слова» и «Запись байта»; «Чтение слова»; «Чтение слова — модификация — запись слова» в ту же ячейку и «Чтение слова — модификация — запись байта».

Операции считывания производят передачу информации от исполнителя задатчику, а операции записи — в обратном направлении. Вид операции задается комбинацией управляющих сигналов, которые задатчик посылает исполнителю.

Рассмотрим процедуру выполнения отдельных операций в интерфейсе «Q-шина». Операция может начаться (шина захвачена) только после снятия сигнала *SYNC* предыдущей операции. Захват шины периферийным устройством осуществляется через систему прерывания и арбитражное запросов прерывания с учетом уровней приоритетов, присвоенных отдельным ПУ. Функционирование системы прерывания и арбитража описано в гл. 9. Здесь ограничимся рассмотрением протокола и временной диаграммы процедуры прерывания и арбитража.

*Протокол и временная диаграмма (рис. 11.24) процедуры прерывания и арбитража.*

1. ПУ выставляет запрос прерывания (ЗП) *IRQ*.

2. Процессор (схема арбитража), восприняв ЗП, в ответ выставляет сигнал *строб данных DIN* как подтверждение приема ЗП, производит арбитражное запросов и (после паузы, равной или более 150 нс) выдает сигнал разрешения прерывания (РП) *IAK*.

3. Первое на пути распространения сигнала *IAK* ПУ, выставившее запрос прерывания, блокирует его дальнейшее распространение и (при условии получения сигнала *DIN*) снимает

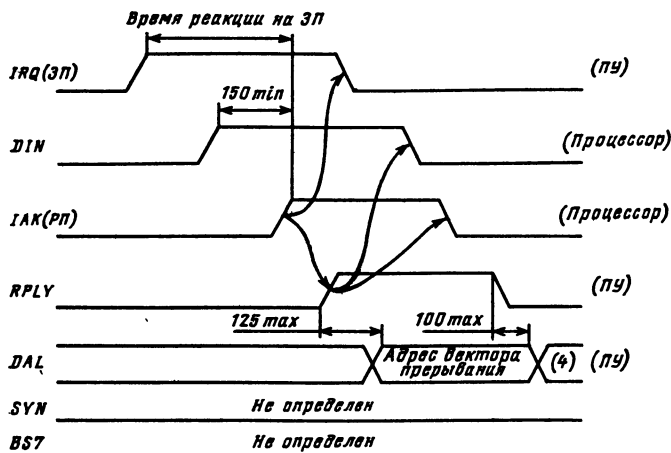


Рис. 11.24. Временная диаграмма процедуры прерывания и арбитража в интерфейсе «Q-шина» (время указано в наносекундах)

сигнал *IRQ*, посылает сигнал *ответ RPLY*, становясь задатчиком, и затем выставляет на линиях *DAL* адрес вектора прерывания.

4. Процессор принимает адрес вектора прерывания, последовательно снимает сигналы *DIN* и *IAK*.

5. ПУ снимает с линий *DAL* адрес вектора прерывания.

6. Процессор реализует прерывание текущей программы и переход к прерывающей программе.

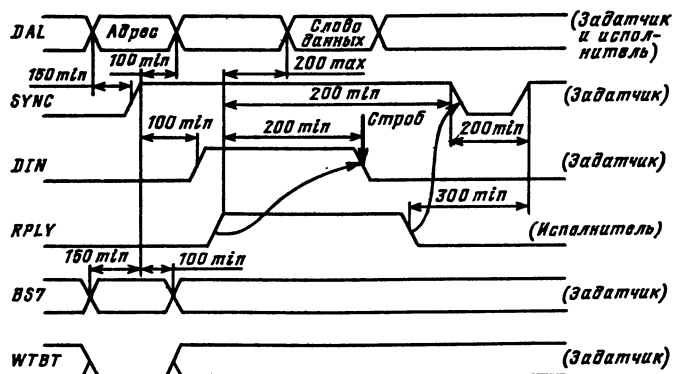


Рис. 11.25. Временная диаграмма операции «чтение слова» в интерфейсе «Q-шина» (время указано в наносекундах)

*Протокол и временная диаграмма (рис. 11.25) операции «Чтение слова»* (задатчиком является процессор или периферийное устройство, исполнителем — память или периферийное устройство).

1. Задатчик после освобождения интерфейса (снят сигнал *SYNC* предыдущей операции) выставляет: на линиях *DAL*  $[21 \div 0]$  адрес (указывает ячейку ОП или регистр ПУ), сигнал *BS7*, если исполнитель — периферийное устройство.

2. Задатчик с задержкой не менее 150 нс (и не раньше чем через 200 нс после снятия *SYNC* и 300 нс после снятия *RPLY* от предыдущей передачи) выставляет сигнал *SYNC*, указывающий всем устройствам, присоединенным к интерфейсу, что интерфейс занят и на его линиях выставлен адрес и задан тип операции. Задержка 75 нс необходима из-за перекоса — разброса моментов времени поступления сигналов по отдельным линиям адреса и еще 75 нс из-за разброса времени декодирования.

3. Устройство, обнаружившее на шине свой адрес, воспринимает предписанный ему тип операции (отсутствие сигнала *WTBT* указывает на операцию считывания) и становится исполнителем: фиксирует в своей аппаратуре адрес и значения сигналов *BS7* и *WTBT*, декодирует адрес.

4. Задатчик с задержкой не менее 100 нс после выставления сигнала *SINC* снимает адрес с линий *DAL* и сигнал *BS7* и выставляет сигнал *DIN*.

5. Исполнитель в ответ на сигнал *DIN* выставляет сигнал *RPLY* и после этого не более чем через 125 нс считывает и выставляет на линии *DAL*  $[15 \div 0]$  запрашиваемое слово, а на линии *DAL*  $[17 \div 16]$  — результат контроля памяти по четности.

6. Задатчик после поступления сигнала *RPLY* с задержкой, равной или большей 200 нс (для компенсации перекоса), принимает (стробирует) информацию с линий *DAL* и снимает сигнал *DIN*.

7. Исполнитель, восприняв сброс сигнала *DIN*, снимает сигнал *RPLY* и после этого не более чем через 100 нс — информацию с линий *DAL*. Этим завершается отключение исполнителя от интерфейса.

8. Задатчик в ответ на снятие сигнала *RPLY* снимает сигнал *SYNC*, освобождая интерфейс.

Процедуры остальных операций имеют много общего с рассмотренной. Так, например, пп. 1 и 2 («адресная» часть процедуры) у всех операций одна и та же.

*Протокол операций «Запись слова» и «Запись байта» (АТО и АТОВ).*

Пункты 1—3 те же, что и в предыдущей процедуре (в п. 1 выставляется дополнительно сигнал *WTBT*).

4. Задатчик с задержкой не менее 100 нс после выставления сигнала *SYNC* снимает адрес с линий *DAL*, сигналы *BS7* и *WTBT* (при записи слова), выставляет слово (или байт) на линиях *DAL* и выдает сигнал *DOUT*.

5. Исполнитель, восприняв сигнал *DOUT*, принимает информацию и выдает сигнал *RPLY*.

6. Задатчик с задержкой не менее 150 нс снимает сигнал *DOUT* и *WTBT* (при записи байта) и после этого с задержкой не менее 100 нс снимает информацию с линий *DAL*.

7. Исполнитель в ответ на снятие сигнала *DOUT* с задержкой не менее 175 нс снимает сигнал *RPLY* и тем самым логически отключается от интерфейса.

8. Задатчик в ответ на снятие сигнала *RPLY* с задержкой не менее 175 нс снимает сигнал *SYNC*, освобождая интерфейс.

Процедуры операций «Чтение слова — модификация — запись слова» и «Чтение слова — модификация — запись байта» повторяют пп. 1—7 процедуры операции «Чтение слова», после чего сигнал *SYNC* не снимается. Задатчик с задержкой не менее 200 нс после снятия сигнала *RPLY* (п. 7 процедуры операции «Чтение слова») выставляет сигнал *DOUT*, и совершается последовательность действий, соответствующих операций «Записи слова» (байта).

*Организация прямого доступа к памяти в интерфейсе «Q-шина».* Прямой (без участия процессора) доступ к памяти (ПДП) используется для непосредственного обмена информацией между ОП и ПУ с блочной передачей, главным образом ЗУ на магнитных дисках.

Прямой доступ к памяти осуществляется под управлением предварительно настраиваемого процессором контроллера ПДП. При настройке процессор загружает в регистры контроллера начальный адрес в памяти для передаваемого блока данных, длину блока (число байт) и вид операции (запись или чтение). Сама операция передачи блока происходит практически без участия процессора.

Схема цепей запроса и разрешения ПДП в интерфейсе «Q-шина» представлена на рис. 11.23. Запросам ПДП присваивается наивысший приоритет на прерывание процессора — более высокий, чем седьмой уровень приоритета в системе прерывания. Таким образом, запросы ПДП всегда имеют более высокий приоритет, чем программа, обрабатываемая процессором.

Мультиплексируемая шина адреса/данных вносит ряд особых обстоятельств в реализацию ПДП в интерфейсе «Q-шина», в силу чего оказывается целесообразным реализовать несколько режимов ПДП в этом интерфейсе.

Передаче каждого слова из памяти или в память должна

предшествовать передаче в память адреса соответствующей ячейки, что при общих линиях адреса и данных заставляет их передавать последовательно во времени, снижая почти вдвое пропускную способность шины. При передаче блока данных обычно используются последовательно расположенные ячейки памяти. В таком случае в память можно передавать только начальный адрес блока данных, а формирование текущего адреса данных возложить на устройство управления памятью. Тогда отпадает необходимость предварять передачу каждого слова передачей его адреса в памяти.

Возникает также проблема недопущения длительной монополизации интерфейса одним контроллером ПДП при наличии запросов от других устройств. С учетом сказанного в интерфейсе «Q-шина» могут (при соответствующем исполнении контроллеров ПДП и устройств ОП) реализовываться следующие режимы ПДП [72].

*Одноцикловый.* После каждого цикла передачи слова снова производится арбитражное разрешение всех запросов ПДП. Теоретическая скорость передачи 1,66 Мбайт/с. На практике реализуется обмен через интерфейс со скоростью до 500 кбайт/с.

*Монопольный.* Арбитражное разрешение производится только 1 раз, а затем контроллер, захватив интерфейс, организует передачу последовательностей адрес — слово данных до завершения передачи блока. Теоретическая скорость передачи сохраняется как в одноцикловом режиме, а практически реализуется передача со скоростью до 1 Мбайт/с. При этом режиме другие устройства, включая процессор, не имеют доступа к шине.

*Ограниченно-монопольный.* В монопольном режиме производится передача только четырех слов, после чего снова производится арбитражное разрешение всех запросов ПДП. Этот режим получил наибольшее распространение.

*Блочный.* Передается начальный адрес, затем подряд (без передачи адресов, которые формирует само устройство памяти) передаются семь слов, после чего опрашивается линия *DMR* на наличие запросов ПДП. При наличии запроса передается восьмое слово и инициируется новое арбитражное разрешение, при отсутствии его передача продолжается до тех пор, пока не будут переданы 16 слов, после чего производится новое арбитражное разрешение запросов. В блочном ПДП за счет уменьшения числа передач адресов и уменьшения квитирований («рукопожатий») при обмене сигналами в интерфейсе теоретическая пропускная способность интерфейса увеличивается почти в 2 раза по сравнению с монопольным режимом. Практически скорость передачи определяется скоростными характеристиками ПУ, участвующего в операции ПДП.



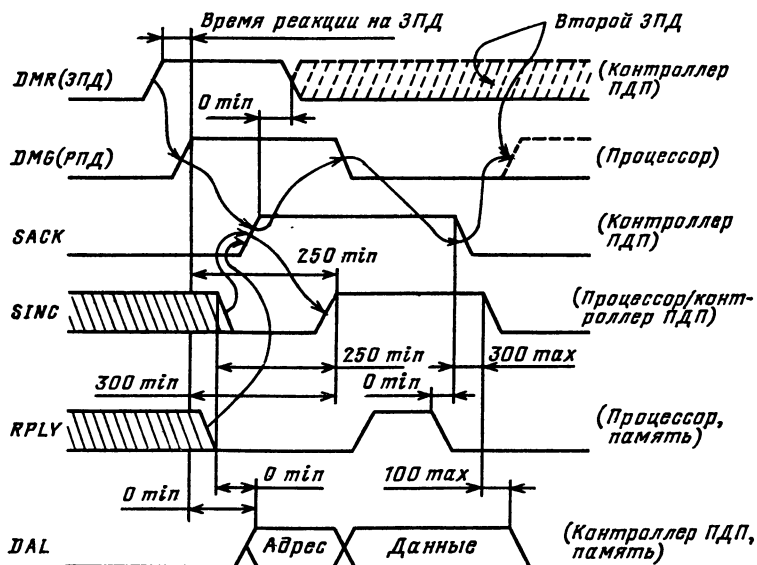


Рис. 11.26. Временная диаграмма прямого доступа к памяти в интерфейсе «Q-шина» (время указано в наносекундах)

Протокол процедуры и временная диаграмма (рис. 11.26) прямого доступа к памяти (задатчик — контроллер ПДП, исполнитель — память).

1. Контроллер ПДП выставляет запрос прямого доступа (ЗПД) *DMR*.

2. Процессор, обнаружив наличие запросов, выставляет сигнал разрешения прямого доступа (РПД) *DMG*, запрещает новый цикл процессора, снимает сигналы *SYNC* и *RPLY*.

3. Первый по пути сигнала *DMG* выставивший запрос контроллер ПДП запрещает его дальнейшее распространение, после снятия сигналов *SYNC* и *RPLY* от предыдущей операции выставляет сигнал подтверждения ПДП *SACK* (сохраняется на время ПДП-передачи) и снимает запрос *DMR*. Контроллер становится задатчиком.

4. Процессор в ответ на сигнал *SACK* снимает сигнал *DMG* и ожидает окончания ПДП-передачи.

5. Контроллер выставляет на линии *DAL* адрес ячейки памяти, выставляет сигнал *SYNC* и затем производится передача четырех слов, выполняемая согласно процедурам операций «Чтение слова» или «Запись слова», после чего производится новое арбитражное запросов.

С увеличением быстродействия процессоров повышаются требования к скорости обмена информацией между ним и ОП. Поэтому новые модификации «Q-шины» предусматривают возможность быстрого доступа процессора к памяти через дополнительное межсоединение с «собственной памятью» [72].

### **11.11. Интерфейс «мультишина» (И-41) микропроцессоров и микроЭВМ**

Интерфейс «мультишина» (И-41), аналогичный интерфейсу Multibus фирмы Intel (США), широко применяется в микроЭВМ, устройствах и системах, в которых используются микропроцессоры K580 и K1810.

Обмен данными осуществляется асинхронно по схеме «задатчик — исполнитель». Интерфейс «мультишина» может работать как с 8-, так и с 16-разрядными микропроцессорами; имеются средства расширения адресного пространства интерфейса. Анализ приоритетов запросов прямого доступа к памяти и запросов прерывания для программно-управляемого (процессорно-управляемого) обмена данными производится отдельными устройствами (соответственно «арбитром» и «блоком приоритетного прерывания»), благодаря чему достигаются упрощение и ускорение процедур арбитража запросов прямого доступа.

Имеются варианты выполнения схем арбитража и процедур прерывания, которые могут выбираться в зависимости от назначения проектируемого микропроцессорного оборудования.

На рис. 11.27 представлена упрощенная структура интерфейса «мультишина». Всего в интерфейсе 72 линии, которые подразделяются на 3 основные шины: адреса (16 линий для адресации памяти емкостью 64 Кбайт плюс 4 резервные), данных (8 двунаправленных линий плюс 8 резервных, используемых при работе с 16-разрядным микропроцессором), управления (36 линий).

Адрес, выставленный задатчиком на шине адреса, определяет исполнителя, при этом шина адреса может мультипликсироваться на обслуживании той или иной группы устройств и общее рабочее адресное пространство интерфейса расширяется до 128 Кбайт.

В интерфейсе выполняются операции (приказы) «Чтение» и «Запись» (с устройствами памяти), «Ввод» и «Вывод» (с периферийными устройствами).

В шине управления можно выделить несколько подшин и отдельных функциональных линий.

*Линия и сигнал инициализации* используются для установки начального состояния.

*Линия и сигнал управления разрядностью шины данных* (8 разрядов при работе с МП К580 и 16 или 8 при работе с МП К1810).

*Подшина управления передачей данных* содержит линию «Подтверждение», по которой исполнитель посылает одноименный сигнал, и линии «Чтение», «Запись», «Ввод» и «Вывод». Задатчик, выставяя сигнал на одну из этих линий, задает тот или иной приказ. Этот сигнал задатчика, определяющий вид приказа и тем самым указывающий, адресуется устройство памяти или регистры периферийного устройства, используется для мультиплексирования шины данных на работу с устройствами памяти или с периферийными устройствами. Сигнал «Подтверждение» исполнителя и сигнал задатчика, назначающий приказ, участвуют в управлении передачей данных с квитиowaniem.

*Линии и сигналы «Запрет ОП» и «Запрет ПЗУ»* служат для мультиплексирования шины адреса для работы с ОП и ПЗУ.

*Подшина защиты от сбоев и отказов в источнике питания* (пять линий). При сбоях и отказах в системе питания сигналы, передаваемые по этой подшине, инициируют процедуры сохранения информации в памяти и переключения на резервное питание.

*Подшины управления прерыванием и арбитража* на рис. 11.27 выделены из шины управления и показаны соответственно в нижней и верхней частях рисунка.

*Подшина управления прерыванием* содержит восемь линий (уровней) запросов прерывания (ЗП) и линию подтверждения прерывания (число уровней прерывания может увеличиваться путем присоединения дополнительных блоков приоритетного прерывания).

В интерфейсе реализуются две процедуры прерывания для запросов программно-управляемого обмена: с внеинтерфейсным формированием адреса вектора прерывания (формируется блоком приоритетного прерывания БПП) и с векторным прерыванием, при котором источник запроса прерывания выставляет на шине данных адрес своего вектора прерывания.

Первая процедура описана в § 10.3. Пока к уровням (линиям) запросов) подключено по одному источнику запросов прерывания, процедура прерывания производится достаточно быстро. Однако она значительно замедляется при возрастании числа источников запросов и подключения нескольких источников к одному уровню прерывания, так как необходимо программным путем определять источник, выставивший запрос прерывания.

В подобных случаях более быстродействующим оказывается векторное прерывание, при котором используются специальный блок прерываний (контроллер прерывания) и показанные на рис. 11.27 прерывистыми линиями дополнительные связи.



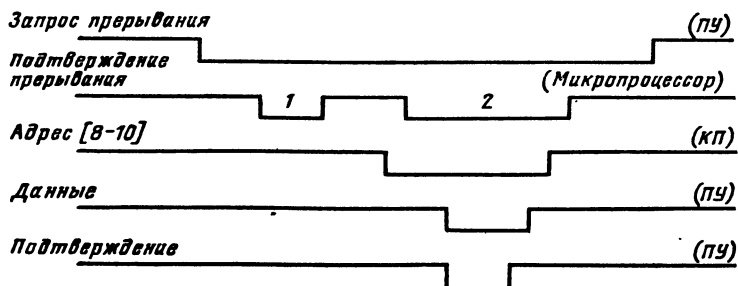


Рис. 11.28. Временная диаграмма векторного прерывания в интерфейсе «мультишина»

Процедура арбитража производится по отношению к запросам прямого доступа памяти и выполняется схемой, называемой арбитром. В рассматриваемом интерфейсе реализуются два варианта арбитража: параллельный и последовательный.

В процедурах параллельного арбитража, схема которого представлена в верхней части рис. 11.27, используются восемь линий запросов прямого доступа (ЗПД) и соответственно восемь линий разрешения прямого доступа (РПД).

При арбитраже каждое устройство выставляет ЗПД на собственную линию запроса. Арбитр сравнивает приоритеты поступающих запросов и выдает сигнал РПД самому приоритетному устройству по индивидуальной линии. В этом случае время выполнения процедуры арбитража может быть существенно уменьшено, так как оно определяется только скоростью работы арбитра и не зависит от числа ПУ. Вместе с тем при этом способе организации арбитража требуются дополнительные аппаратные затраты и ограничивается число устройств в системе числом входов и выходов схемы арбитра.

Более просто реализуется «последовательный арбитраж» (рис. 11.29). В этой процедуре сигналы запросов прямого доступа отсутствуют и становятся ненужными не только линии запросов, но и сам арбитр.

На входе самого приоритетного ПУ всегда присутствует *Сигнал разрешения*, который последовательно транслируется ко входам всех устройств, подключенных к интерфейсу. Порядок подключения определяет приоритет устройства. Обычно микропроцессор имеет самый низкий приоритет.

Любое ПУ может занять интерфейс, если на его входе присутствует «Сигнал разрешения», при этом оно сбрасывает «Выходной сигнал разрешения», запрещая тем самым устройствам с более низким приоритетом запрашивать прямой доступ. Число

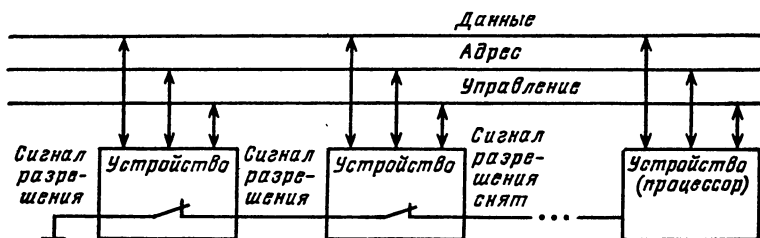


Рис. 11.29. Последовательный арбитраж

подключаемых ПУ ограничивается временем распространения сигнала по линии разрешения прямого доступа.

Интерфейс «мультишина» позволяет воспользоваться и компромиссным решением, так называемым последовательно-параллельным арбитражем, при котором к линии запросов  $i$ -го уровня схемы параллельного арбитража подключаются  $k$  устройств, соединенных между собой по схеме последовательного арбитража.

Иногда требуется смена уровня приоритета устройства в процессе работы. Одним из простых способов динамического изменения приоритета является циклическая смена уровней приоритетов устройств после каждого обращения к интерфейсу.

В интерфейсе «Q-шина» процедура арбитража выполняется асинхронно. Это позволяет удалять ПУ на значительные расстояния, но существенно снижает быстродействие системы прерывания. В интерфейсе «мультишина» процедура арбитража синхронизируется синхросигналами.

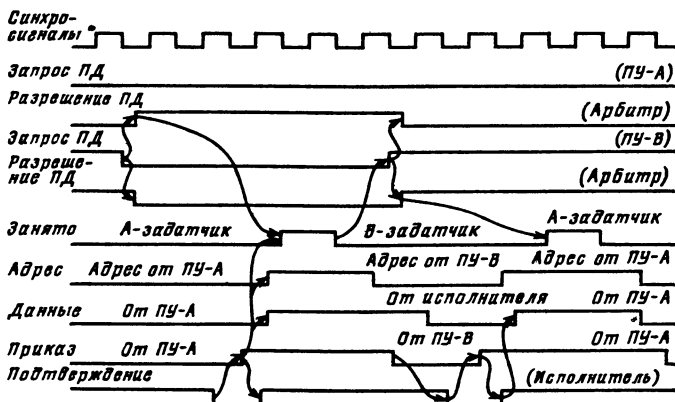


Рис. 11.30. Временная диаграмма смены задатчика и обмена данными в интерфейсе «мультишина» (параллельный арбитраж)

Интерфейс «мультишина» рассчитан на микромашины с короткими межмодульными связями. Это позволяет выполнять смену задатчика на шине синхронно со специальными синхросигналами. На рис. 11.30 приведена временная диаграмма обмена данными в интерфейсе «мультишина» в режиме прямого доступа при параллельном арбитраже.

В исходном состоянии устройство *A* занимает интерфейс. Затем устройство *B*, имеющее более высокий приоритет, запрашивает прямой доступ, выставляя по срезу синхросигнала сигнал запроса прямого доступа на индивидуальную линию запроса. Поскольку устройство *B* имеет более высокий приоритет, чем устройство *A*, арбитр до следующего фронта синхросигнала выдает устройству *B* сигнал *Разрешение ПД*, сбрасывая одновременно сигнал *Разрешение ПД* на входе устройства *A*. По сбросу сигнала *Разрешение* устройство *A* должно завершить выполнение приказа (на рис. 11.30 устройство *A* выполняет приказ *Запись* для текущего байта), при этом сбрасываются сигнал, задающий приказ, адрес исполнителя на шине адреса, данные на шине данных. По ближайшему срезу синхросигнала устройство *A* освобождает интерфейс, сбрасывая сигнал *Занято*, а по следующему срезу синхросигнала устройство *B*, выставляя сигнал *Занято*, занимает интерфейс, становится задатчиком и выдает приказ *Чтение*.

## 11.12. Особенности интерфейса «мультишина-II» (Multibus II)

Быстрое увеличение производительности, расширение логических возможностей микропроцессоров и еще более быстрый рост требований пользователей к вычислительной мощности создаваемых на их основе микроЭВМ и микропроцессорных систем (МП-систем) потребовали развития новых подходов в построении интерфейсов МП-систем. Попробуем восстановить логику выработки новых архитектурных решений в этой области.

Микропроцессоры по своей производительности приблизились к малым ЭВМ, но из-за сравнительно низкой пропускной способности систем ввода-вывода (несколько сотен килобайт в секунду) их нельзя было использовать в оборудовании таких массовых изделий, как мощные автоматизированные рабочие места (АРМ). Необходимо было увеличить пропускную способность интерфейса до нескольких десятков мегабайт в секунду, что побуждало к переходу от асинхронного способа передачи данных (см. § 11.11) к синхронному, хотя такое решение связано с существенным уменьшением (до 1 м) допустимого расстояния между отдельными модулями системы.

Однако для многих применений даже возросшая производительность МП оказалась недостаточной, что закономерно с учетом малой стоимости МП и БИС памяти привело к созданию микропроцессорных (микромашинных) систем (см. гл. 15).

Это наложило специальные требования на организацию интерфейса МП-систем. Интерфейс должен позволять объединить в МП-систему несколько модулей микроЭВМ и (или) микропроцессорных устройств без существенных потерь в пропускной способности из-за конфликтов (столкновений) при их попытках использовать интерфейс. Необходимо иметь возможность высокоскоростного межмодульного перемещения данных и межмодульных прерываний с малым временем реакции, в частности, для реализации межзадачных связей при многозадачных режимах обработки данных. При этом приходится отказываться от принятого во многих интерфейсах арбитражного запросов прерывания, поступающих по нескольким линиям, которым присвоены разные уровни приоритета. Таких линий в микропроцессорной системе оказалось бы  $n(n-1)k$  ( $n$  — число МП,  $k$  — количество уровней приоритета), что делает такой подход для многих МП-систем практически нереализуемым. Возникшие проблемы разрешимы путем отказа от обычной для многих интерфейсов передачи информационной и управляющей информации в форме сигналов и перехода к применению метода передачи сообщений при межмодульном обмене данными и межмодульных прерываниях.

Во многих микропроцессорных системах возникает необходимость реализации территориально в большей или меньшей степени распределенной обработки данных, что приводит к необходимости (особенно в интерфейсах с синхронной передачей, ограничивающей расстояния между модулями системы) иметь в составе систем ввода-вывода средства дистанционного обмена данными между модулями по последовательному каналу, другими словами, иметь средства реализации локальной вычислительной сети (см. гл. 16).

Наконец, система ввода-вывода должна сохранять возможность выполнения в микроЭВМ обычных операций ввода-вывода с периферийными устройствами, в том числе в режиме прямого доступа к памяти.

Одним из ориентированных на микропроцессорные системы является процессорно-независимый интерфейс Multibus II («мультишина II») фирмы Intel (США) [25, 80].

Интерфейс Multibus II практически представляет собой комплекс из пяти совместимых интерфейсов, которые могут использоваться в МП-системе в различных комбинациях:



а) параллельный интерфейс с синхронной передачей сообщений, а также 8-, 16- или 32-разрядных слов по совмещенной шине адреса/данных (при числе абонентов шины до 20) с частотой сигналов по линии интерфейса 10 МГц и суммарной пропускной способностью (при использовании 32-разрядной ширины интерфейса) до 40 Мбайт/с (при передаче сообщение разбивается на пакеты длиной 32 байта, а для повышения пропускной способности интерфейса используется мультиплексирование на шине пакетов от разных задатчиков);

б) последовательный интерфейс со скоростью передачи последовательным кодом сообщений 2 Мбит/с, позволяющий реализовывать локальную сеть со случайным доступом по протоколу «множественный доступ к каналу с контролем несущей и обнаружением конфликтов» (CSMA/CD) (см. гл. 16), при максимальном расстоянии между абонентами до 10 м и числе ответвлений до 32 на 10-метровой длине линии (канала) для подключения абонентов (микроЭВМ и других устройств);

в) локальная шина для межустройственного в микроЭВМ обмена с квитированием 8-, 16- или 32-разрядными словами или блоками с частотой передачи сигналов в шине 12 МГц и максимальной суммарной пропускной способностью до 48 Мбайт/с;

г) интерфейс прямого доступа к памяти;

д) расширенная шина ввода-вывода.

Первые три интерфейса определяются протоколом (стандартом) Multibus II, два последних — протоколом Multibus I (И-41).

Проектировщик МП-системы в зависимости от ее назначения и особенностей структуры принимает решение о составе используемых интерфейсов.

На рис. 11.31 представлена многомикропроцессорная система, в которой применены все пять типов интерфейсов.

Объединяемые интерфейсом микроЭВМ и специализированные микропроцессорные устройства содержат интерфейсные БИС, в том числе «сопроцессоры передачи сообщений» (СПС), расположенные на каждой плате, подключенной к параллельному интерфейсу. Сопроцессоры передачи сообщений задатчика и исполнителя, взаимодействуя друг с другом, осуществляют непосредственное управление передачей сообщений через шину интерфейса, включая разбиение сообщений на 32-байтные пакеты, передачу пакетов, сборку пакетов в сообщение, буферизацию передаваемых данных для согласования скоростей передачи данных в параллельном интерфейсе и локальной шине. Сопроцессоры передачи сообщений освобождают главные (обрабатывающие информацию) МП от необходимости ожидать доступ к шине и от управления передачей данных через шину.

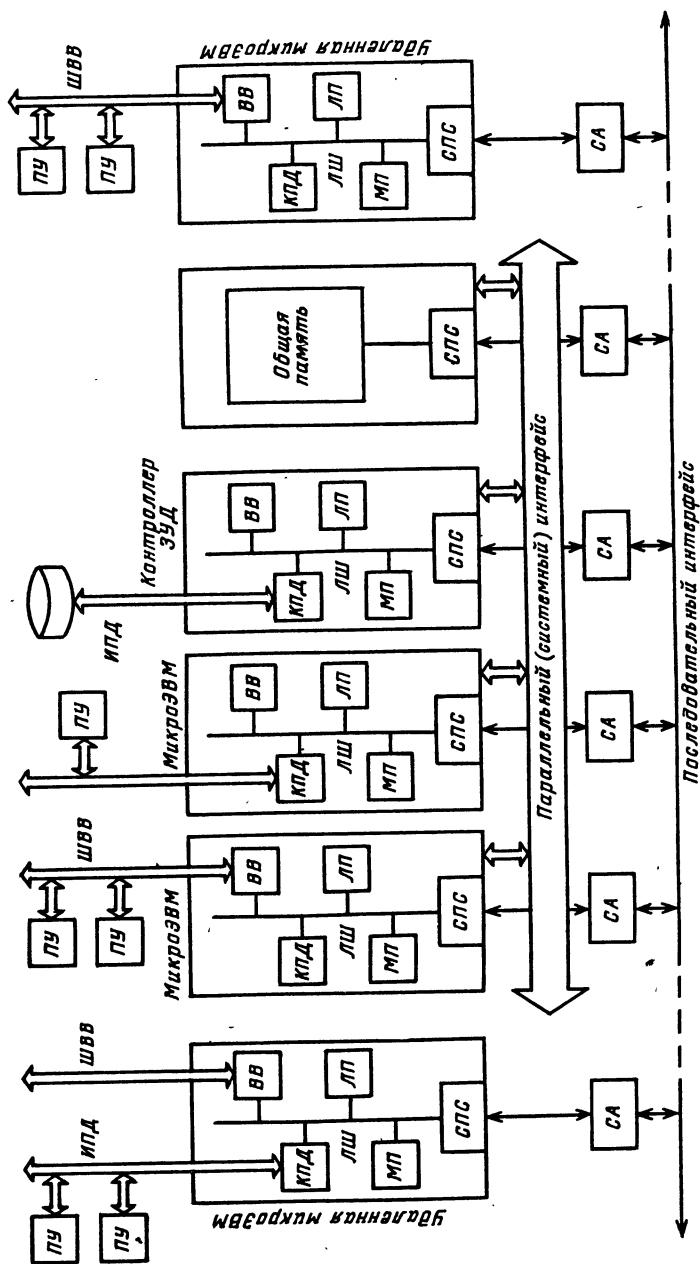


Рис. 11.31. Структура МП-системы на основе интерфейса Multibus II

Наличие в объединяемых интерфейсом микроЭВМ локальных памятей, подключенных к МП через локальную шину, позволяет параллельно во времени производить выборку команд и передачу данных через параллельный интерфейс, что увеличивает производительность микроЭВМ примерно в 1,5—2 раза.

К параллельному интерфейсу подключаются общая разделяемая память и общие ПУ, причем последние — через интерфейс прямого доступа. Схемы и процедуры арбитража запросов прерывания на шине ввода-вывода и в интерфейсе прямого доступа определяются стандартом Multibus I (И-41) (см. § 11.11).

Межмодульные прерывания в параллельном интерфейсе реализуются в форме «виртуального прерывания», при котором запрашивающий прерывание модуль посылает через шину прерываемому модулю специальное «прерывающее сообщение», оформленное в виде 32-байтного пакета, содержащего 8-разрядные номера источника и адресата сообщения, 8-разрядный указатель специального характера сообщения и до 28 байт данных, конкретизирующих запрашиваемое прерывание (адрес его вектора прерывания и другие сведения). Время передачи этого пакета при пропускной способности интерфейса 40 Мбайт/с составляет 1 мкс. Отметим, что виртуальное прерывание можно рассматривать как развитие векторного прерывания (см. гл. 9).

Передаче «прерывающего сообщения» должно предшествовать занятие интерфейса запрашивающим прерывание устройством. В параллельном интерфейсе реализован децентрализованный арбитраж запросов на занятие интерфейса. Для этого используются коллективные «линия запроса» и «линии приоритета».

Устройства, пытающиеся занять интерфейс, выставляют запрос на линию запроса и присвоенные им приоритетные номера на линии приоритетов. Соответствующие схемы обеспечивают сохранение на этих линиях только номера наиболее приоритетного устройства. Право занятия интерфейса получает устройство, опознавшее свой приоритетный номер на линиях приоритета.

### **Контрольные вопросы**

1. В чем заключаются проблемы организации системы ввода-вывода ЭВМ и каковы пути их решения? Какое значение при этом имеет унификация команд ввода-вывода, форматов данных, интерфейсов?

2. Что такое прямой доступ к памяти? Какие функции возлагаются на контроллер прямого доступа к памяти?

3. Сравните интерфейсы, используемые в малых ЭВМ и микропроцессорах, с интерфейсом ввода-вывода в ЕС ЭВМ?

4. Зачем на канал ввода-вывода возлагают реализацию цепочек данных и цепочек операций?

5. Сравните байт-мультиплексный, селекторный и блок-мультиплексный каналы.

6. Каковы функции буферов в системах ввода-вывода? Приведите примеры виртуального изменения буфером данных количественных и качественных характеристик периферийного устройства.

7. Что общего и в чем различия интерфейсов «Q-шина» и «мультишина» (И-41)?

## Глава 12

# СИСТЕМЫ АВТОМАТИЧЕСКОГО КОНТРОЛЯ И ДИАГНОСТИРОВАНИЯ ЭВМ

## 12.1. Основные характеристики надежности ЭВМ. Функции систем контроля и диагностирования

Согласно современным представлениям под надежностью понимается свойство изделия (элемента, узла, устройства, машины, системы) выполнять заданные функции, сохранять свои характеристики в установленных пределах при определенных условиях эксплуатации.

Надежность вычислительной машины определяется безотказностью, достоверностью функционирования и ремонтпригодностью.

*Безотказность* есть свойство машины или системы, характеризующееся закономерностями возникновения отказов. Под отказом понимается событие, заключающееся в полной или частичной утрате машиной (системой) работоспособности. Отказ ЭВМ — это такое нарушение ее работоспособности, для восстановления которой требуются определенные действия обслуживающего персонала по ремонту, замене и регулировке неисправного элемента, узла или устройства. Безотказность может оцениваться *средним временем наработки машины на один отказ*.

*Ремонтпригодность* есть степень приспособленности машины или системы к предупреждению, обнаружению и устранению отказов. Ремонтпригодность, определяя потерю работоспособности машины вследствие необходимости производить устранение неисправностей, характеризуется *средним временем устранения неисправности*.

Работа ЭВМ заключается в выполнении преобразований информации, основными из которых являются передача информа-

ции в пространстве (между отдельными блоками и устройствами машины), хранение информации (передача информации во времени), арифметические и логические преобразования.

В силу указанной специфики рабочего процесса надежность ЭВМ наряду с безотказностью определяется также достоверностью функционирования. *Достоверность функционирования* есть свойство машины (системы), определяемое безошибочностью производимых машиной (системой) преобразований информации и характеризующееся закономерностями появления ошибок из-за сбоев.

*Сбоем* называют кратковременное самоустранившееся нарушение нормального функционирования машины вследствие кратковременного воздействия на некоторый элемент (или элементы) внешних помех или изменений некоторых других входных воздействий, а также из-за кратковременного изменения параметров элементов (кратковременные нарушения контактов и т. п.). После сбоя машина длительное время может работать нормально. Сбой сопровождается искажением информации при операциях передачи, хранения или обработки ее. Следовательно, если не устранить последствия сбоя, то задача может оказаться неправильно решенной из-за искажений в данных, промежуточных результатах или в самой программе.

Однако если при отказе для восстановления работоспособности машины или системы необходимо устранить неисправность в аппаратуре, то при сбое требуется восстановить лишь достоверность информации, что хотя и связано с потерями рабочего времени ЭВМ (например, на повторный пуск программы или ее части), но не требует ремонта или регулировки аппаратуры. В силу этого восстановление достоверности функционирования сравнительно легко может быть автоматизировано.

Достоверность функционирования ЭВМ можно оценить *средним временем наработки машины на один сбой*. Для более полной оценки достоверности функционирования введем в состав характеристик надежности ЭВМ *среднее время восстановления достоверности информации после сбоя*.

В процессе развития вычислительной техники повышение безотказности ЭВМ достигалось использованием более надежных элементов (переход от ламповых схем к полупроводниковым и затем к интегральным микросхемам), применением облегченных нагрузочных режимов для схемных компонентов, совершенствованием конструкции и технологии (печатный, в том числе многослойный монтаж, улучшение вентиляции блоков машин, автоматизация проектирования, изготовления и контроля узлов и др.) и соответствующими логическими решениями (см. § 15.4).

Для уменьшения вероятности сбоев принимаются меры для уменьшения помех в цепях электронных схем: согласование нагрузок электронных схем, специальные методы монтажа и выполнения заземлений схем.

Пользователь должен быть уверен в правильности производимых машиной расчетов, особенно при работе ЭВМ в реальном времени с выдачей управляющих воздействий на объект управления.

Если вычислительная машина не обеспечивает пользователя средствами контроля достоверности ее функционирования, он вынужден непроизводительно затрачивать машинное время на двойной просчет, решение контрольных вариантов и т. д.

Выбор методов повышения надежности ЭВМ и их эффективность в значительной мере зависят от того, является вычислительная машина восстанавливаемой или невосстанавливаемой, обслуживаемой или необслуживаемой системой.

Система называется *восстанавливаемой*, если во время эксплуатации может производиться ремонт для устранения возникающих отказов. Система считается *обслуживаемой*, если допускается периодическое проведение профилактических испытаний для выявления элементов и узлов, параметры которых близки к предельно допустимым.

Целью профилактических испытаний является увеличение среднего времени наработки на отказ в период между профилактическими работами.

В большинстве случаев ЭВМ представляют собой в терминах теории надежности восстанавливаемые обслуживаемые системы. К невосстанавливаемым системам относятся, например, бортовые вычислительные машины ракет.

Профилактическое обслуживание связано с потерями рабочего времени машины и затратами труда обслуживающего персонала. Эти потери по своему характеру близки к потерям, связанным с устранением отказов, и они должны учитываться наряду с показателями надежности устройств машины при назначении периодов и объема профилактического обслуживания.

Очевидно, что чем выше надежность ЭВМ, тем больше может быть период между профилактическими работами. Чем меньше тратится машинного времени и квалифицированного труда на профилактические работы и устранение неисправностей, тем выше *обслуживаемость* ЭВМ, т. е. степень приспособленности машины к процессам обслуживания.

В целях оценки совокупного влияния на работу ЭВМ рассмотренных выше отдельных показателей надежности введем *комплексный коэффициент эксплуатационной надежности* (ком-

плексный коэффициент использования)

$$K_n = \frac{\sum_{i=1}^n t_i - \sum_{r=1}^m \tau_{в.ср} - \tau_k - \sum_{s=1}^e \tau_{пфs}}{\sum_{i=1}^n t_i + \sum_{j=1}^n \tau_{в.ож} + \sum_{j=1}^n \tau_{ожj}}, \quad (12.1)$$

где  $n$ ,  $m$ ,  $e$  — соответственно число отказов, сбоев и профилактических обслуживаний за достаточно большой период эксплуатации;  $t_i$  — интервал времени исправной работы машины (системы) между  $(i-1)$ -м и  $i$ -м нарушениями функционирования машины из-за отказов;  $\tau_{в.ож}$  — время восстановления после  $j$ -го отказа;  $\tau_{в.ср}$  — время восстановления достоверности информации после  $r$ -го сбоя (время, потраченное на повторный пуск программы, части программы, команды и т. д.);  $\tau_k$  — суммарное машинное время, затраченное в рассматриваемый период на контроль достоверности (на двойной расчет, контрольные варианты, работу схем контроля и т. д.);  $\tau_{пфs}$  — время, затраченное на  $s$ -е профилактическое обслуживание;  $\tau_{ожj}$  — время ожидания начала ремонта после  $j$ -го отказа.

В знаменателе (12.1) находится общее время наблюдения за работой ЭВМ. Это время включает в себя наряду с суммарным временем работоспособного состояния машины суммарное время ожидания начала ремонта после отказов и продолжительности ремонта. В числителе находится общее время полезной работы машины.

Приведенное выражение позволяет сделать важные выводы.

Для повышения комплексного коэффициента использования необходимо повышать обслуживаемость машины и добиваться уменьшения потерь времени на устранение отказов (повышение ремонтпригодности). Эти потери времени в таких сложных объектах, как ЭВМ, в первую очередь связаны с поиском места неисправности. Важнейшим средством уменьшения указанных потерь и повышения обслуживаемости ЭВМ является *система автоматического диагностирования*, позволяющая локализовать неисправность.

Чтобы уменьшить потери от сбоев и отказов, порождающих ошибки, надо предотвратить распространение ошибки в вычислительном процессе, так как в противном случае существенно усложнятся и удлинятся процедуры проверки правильности работы программы, определения и устранения искажений в программе, данных и промежуточных результатах.

Для этого необходимо обнаруживать появление ошибки в выполняемых машиной преобразованиях информации возмож-

но ближе к моменту ее возникновения. С этой целью надо иметь *систему автоматического контроля* правильности работы ЭВМ, которая при появлении ошибки в работе машины немедленно приостанавливает выполнение программы. Наличие такой системы освобождает от забот по контролю достоверности и снижает связанные с этим потери [ $\tau_k$  в (12.1)].

Для уменьшения значения второго члена в числителе (12.1) следует иметь *систему автоматического восстановления вычислительного процесса*, распознающую характер (сбой или отказ) ошибки и при сбое автоматически восстанавливающую достоверность информации и выполнение программы, а при отказе иницилирующую работу системы автоматического диагностирования ЭВМ.

Обнаружение ошибок должно производиться в машине непрерывно и, следовательно, не должно вызывать заметного снижения быстродействия машины. Поэтому эта функция возлагается обычно на быстродействующие аппаратные средства контроля, которые позволяют почти полностью совместить во времени выполнение основных и контрольных операций.

Необходимость в коррекции ошибок, восстановлении вычислительного процесса и диагностировании неисправностей при современном уровне надежности ЭВМ возникает достаточно редко. Поэтому целесообразно использовать для выполнения этих функций главным образом микропрограммные, а также программные средства в виде корректирующих и диагностических микропрограмм и программ. Однако чтобы эти программы не были чрезмерно сложны, предусматриваются и определенные аппаратные средства, поддерживающие процедуры восстановления после сбоев и локализации неисправностей.

Основными характеристиками системы автоматического контроля правильности функционирования ЭВМ являются: а) отношение количества оборудования, охваченного системой контроля, к общему количеству оборудования ЭВМ; б) вероятность обнаружения системой контроля ошибок в функционировании ЭВМ; в) степень детализации, с которой система контроля указывает место возникновения ошибки<sup>1</sup>; г) отношение количества оборудования системы контроля к общему количеству оборудования ЭВМ.

Основными характеристиками систем автоматического диагностирования являются: а) вероятность правильного обнаружения места отказа; б) разрешающая способность, равная среднему числу подозреваемых сменных блоков; в) доля аппа-

---

<sup>1</sup> Детальное распознавание места неисправности является задачей системы диагностирования (см. § 12.5).



ратурных средств системы диагностирования в общем оборудовании ЭВМ.

Для снижения затрат машинного времени и труда на профилактические испытания следует снабжать ЭВМ *аппаратурно-программными средствами автоматизации испытаний* [27].

Как следует из (12.1), для повышения коэффициента использования ЭВМ следует уменьшать время ожидания начала ремонта после возникновения отказа, что достигается соответствующими организацией и материально-техническим обеспечением эксплуатационного обслуживания машин, в том числе путем создания для крупных ЭВМ дистанционного обслуживания из специальных центров, связанных с обслуживаемыми машинами средствами телеобработки данных [27].

При создании новых ЭВМ разработка вопросов их технического обслуживания, создания аппаратурно-программных средств повышения надежности функционирования и поддержки эксплуатационного обслуживания ЭВМ, в первую очередь систем автоматического контроля, восстановления и диагностирования, должна вестись параллельно и взаимосвязано с проектированием машины. В современных крупных ЭВМ (например, в ЕС ЭВМ) обнаруживается тенденция к выделению в отдельную систему средств управления и поддержки обслуживания ЭВМ [27, 45].

## 12.2. Контроль передачи информации

При контроле передачи информации наибольшее распространение получили методы *информационной избыточности*, использующие коды с обнаружением и коррекцией ошибок.

Если длина кода  $n$  разрядов, то таким двоичным кодом можно представить максимум  $2^n$  различных слов. Если все разряды слова служат для представления информации, код называется простым (*неизбыточным*). Коды, в которых лишь часть кодовых слов используется для представления информации, называются *избыточными*. Часть слов в избыточных кодах является запрещенной, и появление таких слов при передаче информации свидетельствует о наличии ошибки.

Принадлежность слова к разрешенным или запрещенным словам определяется правилами кодирования, и для различных кодов эти правила различны.

Коды разделяются на равномерные и неравномерные. В равномерных кодах все слова содержат одинаковое число разрядов. В неравномерных кодах число разрядов в словах может быть различным. В вычислительных машинах применяются преимущественно равномерные коды.

Равномерные избыточные коды делятся на разделимые и неразделимые. Разделимые коды всегда содержат постоянное число информационных (т. е. представляющих передаваемую информацию) и избыточных разрядов, причем избыточные занимают одни и те же позиции в кодовом слове. В неразделимых кодах разряды кодового слова невозможно разделить на информационные и избыточные.

Способность кода обнаруживать или исправлять ошибки определяется так называемым минимальным кодовым расстоянием. Кодовым расстоянием между двумя словами называется число разрядов, в которых символы слов не совпадают. Если длина слова  $n$ , то кодовое расстояние может принимать значения от 1 до  $n$ . Минимальным кодовым расстоянием данного кода называется минимальное расстояние между двумя любыми словами в этом коде. Если имеется хотя бы одна пара слов, отличающихся друг от друга только в одном разряде, то минимальное расстояние данного кода равно 1.

Простой (неизбыточный) код имеет минимальное расстояние  $d_{min}=1$ . Для избыточных кодов  $d_{min}>1$ . Если  $d_{min}\geq 2$ , то любые два слова в данном коде отличаются не менее чем в двух разрядах, следовательно, любая одиночная ошибка приведет к появлению запрещенного слова и может быть обнаружена. Если  $d_{min}=3$ , то любая одиночная ошибка создает запрещенное слово, отличающееся от правильного в одном разряде, а от любого другого разрешенного слова — в двух разрядах. Заменяя запрещенное слово ближайшим к нему (в смысле кодового расстояния) разрешенным словом, можно исправить одиночную ошибку.

В общем случае, чтобы избыточный код позволял обнаруживать ошибки кратностью  $r$ , должно выполняться условие

$$d_{min} \geq r + 1. \quad (12.2)$$

Действительно, одновременная ошибка в  $r$  разрядах слова создает новое слово, отстоящее от первого на расстоянии  $r$ . Чтобы оно не совпало с каким-либо другим разрешенным словом, минимальное расстояние между двумя разрешенными словами должно быть хотя бы на единицу больше, чем  $r$ .

Для исправления  $r$ -кратной ошибки необходимо, чтобы новое слово, полученное в результате такой ошибки, не только не совпадало с каким-либо разрешенным словом, но и оставалось ближе к правильному слову, чем к любому другому разрешенному. От правильного слова новое отстоит на расстоянии  $r$ . Следовательно, от любого другого разрешенного слова оно должно отстоять не менее чем на  $r+1$ , а минимальное кодовое расстоя-

ние должно быть не менее суммы этих величин:

$$d_{\min} \geq 2r + 1. \quad (12.3)$$

Код с проверкой четности образуется добавлением к группе информационных разрядов, представляющих простой (неизбыточный) код, одного избыточного (контрольного) разряда.

При формировании кода слова в контрольный разряд записывается 0 или 1 таким образом, чтобы сумма 1 в слове, включая избыточный разряд, была четной (при контроле по четности) или нечетной (при контроле по нечетности). В дальнейшем при всех передачах, включая запись в память и считывание, слово передается вместе со своим контрольным разрядом. Если при передаче информации приемное устройство обнаруживает, что в принятом слове значение контрольного разряда не соответствует четности суммы 1 слова, то это воспринимается как признак ошибки.

Максимальное расстояние кода  $d_{\min} = 2$ , поэтому код с проверкой четности обнаруживает все одиночные ошибки и, кроме того, все случаи нечетного числа ошибок (3, 5 и т. д.). При одновременном возникновении двух или любого другого четного числа ошибок код с проверкой четности не обнаруживает ошибок.

При контроле по нечетности контролируется полное пропадание информации, поскольку кодовое слово, состоящее из 0, относится к запрещенным.

Код с проверкой четности имеет небольшую избыточность и не требует больших затрат оборудования на реализацию контроля. Этот код широко применяется в вычислительных машинах для контроля передач информации между регистрами и считываемой информации в оперативной памяти.

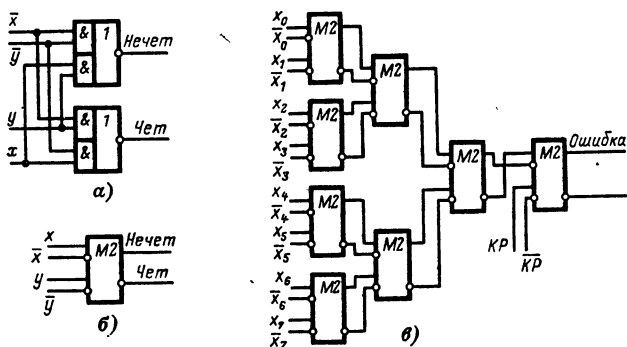


Рис. 12.1. Схемы определения четности

При построении схем определения четности суммы 1 слова используют логические элементы с парафазным выходом, подобным изображенному на рис. 12.1, а и б. Показанные схемы выполняют операцию сложения по модулю 2 (условное обозначение  $M2$ ) для двоичных переменных  $x$  и  $y$ . На рис. 12.1, в показана схема определения признака четности байта.

Каждый информационный символ должен быть задан прямым и инверсным кодами. Структура схемы проверки четности является многоступенчатой, т. е. слово делится на несколько групп разрядов, в каждой из которых проверка четности производится прямым способом (первая ступень), далее производится проверка четности для групп второй ступени, образованных из групп первой ступени, четности которых в этом случае рассматриваются как обычные двоичные разряды, и т. д. до окончательной проверки четности суммы 1 всего слова. В последней ступени четность байта сравнивается со значением контрольного разряда  $KP$ .

Легко установить связь кодирования при контроле по четности с выполнением сложения по модулю 2. Если количество 1 в слове должно быть четным, то в контрольный разряд записывается прямой код суммы по модулю 2 всех информационных разрядов слова. При контроле на нечетность в контрольный разряд заносится обратный код указанной суммы.

*Корректирующий код Хэмминга.* В оперативной памяти применяют код Хэмминга, позволяющий исправлять ошибки.

Код Хэмминга строится таким образом, что к имеющимся информационным разрядам слова добавляется определенное число контрольных разрядов, которые формируются перед записью слова в ОП и вместе с информационными разрядами слова записываются в память.

При считывании слова контрольная аппаратура образует из прочитанных информационных и контрольных разрядов корректирующее число, которое равно 0 при отсутствии ошибки, либо указывает место ошибки, например двоичный порядковый номер ошибочного разряда в слове. Ошибочный разряд автоматически корректируется изменением его состояния на противоположное.

Рассмотрим процесс кодирования для кода Хэмминга с коррекцией одиночной ошибки (минимальное кодовое расстояние  $d_{min}=3$ ).

Если в младшем разряде корректирующего числа появится 1, то это означает ошибку в одном из тех разрядов слова, порядковые номера которых имеют 1 в младшем разряде (т. е. разрядов с нечетными номерами). Введем первый контрольный разряд, которому присвоим нечетный порядковый номер и который установим при кодировании таким образом, чтобы

сумма 1 всех разрядов с нечетными порядковыми номерами была равна 0. Эта операция может быть записана в виде

$$E_1 = x_1 \oplus x_3 \oplus x_5 \oplus \dots = 0,$$

где  $x_1, x_3$  и т. д. — двоичные символы, размещенные в разрядах с порядковыми номерами 1, 3 и т. д.

Появление 1 во втором разряде (справа) корректирующего числа означает ошибку в одном из тех разрядов слова, порядковые номера которых (2, 3, 6, 7, 10, 11, 14, 15 и т. д.) имеют 1 во втором справа разряде. Поэтому вторая операция кодирования, позволяющая найти второй контрольный разряд, имеет вид

$$E_2 = x_2 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_{10} \oplus x_{11} \oplus \dots = 0.$$

Рассуждая аналогичным образом, можно определить все другие контрольные разряды путем выполнения операций

$$E_3 = x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_{12} \oplus x_{13} \oplus x_{14} \oplus x_{15} \oplus \dots = 0;$$

$$E_4 = x_8 \oplus x_9 \oplus x_{10} \oplus x_{11} \oplus x_{12} \oplus x_{13} \oplus x_{14} \oplus x_{15} \oplus x_{24} \oplus \dots = 0$$

и т. д.

После приема кодового слова (совместно со сформированными контрольными разрядами) выполняются те же операции подсчета, которые были описаны выше, а образующееся число

$$E_k E_{k-1} \dots E_3 E_2 E_1$$

считается корректирующим.

При отсутствии ошибок  $E_k E_{k-1} \dots E_2 E_1 = 0$ , при наличии ошибки не равными нулю будут те суммы  $E_i$ , в образовании которых участвовал ошибочный разряд; корректирующее число при этом будет равно порядковому номеру ошибочного разряда.

Выбор места для контрольных разрядов производится таким образом, чтобы контрольные разряды участвовали только в одной операции подсчета четности. Это упрощает процесс кодирования. Рассмотрение выражений для  $E_1, E_2, E_3$  и т. д. показывает, что такими позициями являются разряды с номерами, являющимися целыми степенями двойки: 1, 2, 4, 8, 16 и т. д.

Требуемое число контрольных разрядов (или, что то же самое, разрядность корректирующего числа) определяется из следующих соображений. Пусть кодовое слово длиной  $n$  разрядов имеет  $m$  информационных и  $k = n - m$  контрольных разрядов. Корректирующее число длиной  $k$  разрядов описывает  $2^k$  состояний, соответствующих отсутствию ошибки и появлению ошибок в  $i$ -м разряде. Таким образом, должно соблюдаться

$$2^k \geq n + 1, \quad (12.4)$$

или

$$2^k - k - 1 \geq m. \quad (12.5)$$

Из этого неравенства следует, например, что пять контрольных разрядов позволяют передавать в коде Хэмминга до 26 информационных разрядов.

Если в ОП одновременно записываются или считываются восемь информационных байт (64 разряда), то при использовании кода Хэмминга потребуется семь дополнительных контрольных разрядов.

Контроль по коду Хэмминга реализуется с помощью набора схем подсчета четности (см. рис. 12.1), которые при кодировании определяют контрольные разряды, а при декодировании формируют корректирующее число.

*Модифицированный код Хэмминга.* К контрольным разрядам Хэмминга добавляется еще один (в последнем примере восьмой) разряд КР контроля четности всех одновременно считываемых (записываемых) информационных и контрольных разрядов. При считывании формируются корректирующее число  $E_k E_{k-1} \dots E_1$  и разряд общей четности КР' для всех считанных разрядов, включая КР. Модифицированный код Хэмминга позволяет устранять одиночные и обнаруживать двойные ошибки, как это следует из табл. 12.1.

*Коррекция двойных ошибок в ОП.* При использовании в ОП модифицированного кода Хэмминга может производиться коррекция двойных ошибок.

Пусть  $X$  — слово, записанное в ОП, а  $X'$  — считанное из ОП слово, в котором обнаружены две ошибки. Тогда по сигналу схемы контроля инициируется следующая процедура.

В неисправную ячейку ОП записывается обратный код считанного слова  $\bar{X}'$ , и затем производится его считывание. Над

Т а б л и ц а 12.1

Корректирующее число Хэмминга $E_k E_{k-1} \dots E_1$	КР'	Наличие ошибок
Нули	0	Нет ошибок
$\neq 0$	1	Одиночная ошибка
$\neq 0$	0	Двойная ошибка
Нули	1	Ошибка в контрольном разряде общей четности

получаемым при этом кодом  $(\bar{X}')'$  и кодом  $\bar{X}'$  производится операция

$$Z = \bar{X}' \oplus (\bar{X}')'.$$

Код  $Z$  содержит 1 в разрядах, в которых имеются ошибки.

Схемы управления ОП по коду  $Z$  корректируют одну ошибку. После этого схема коррекции одной ошибки исправляет вторую ошибку.

### 12.3. Контроль арифметических операций

Арифметические операции, как правило, можно представить в виде последовательностей следующих элементарных операций: передача слова и операции преобразования содержимого триггерных регистров — сдвиг, взятие обратного кода и сложение.

Операция сдвига информации в регистре представляет собой, по существу, передачу информации из  $i$ -х разрядов регистра в  $(i+m)$ -е или  $(i-m)$ -е разряды в зависимости от направления сдвига ( $m$  — число разрядов, на которое производится сдвиг). Поэтому для контроля операции сдвига можно использовать те же методы, что и для контроля передачи информации, например контроль четности суммы 1.

Регистр, в котором производится сдвиг, должен иметь дополнительный контрольный разряд, устанавливаемый перед сдвигом в такое состояние, чтобы сумма 1 регистра вместе с контрольным разрядом была, например, четной. Кроме схемы определения общей четности содержимого регистра необходимы схемы, устанавливающие четность разности между числом 1, выдвигаемых из регистра, и числом 1, вдвигаемых в регистр. Если в освобождающиеся при сдвиге разряды вдвигаются 0, то достаточно иметь схему определения четности суммы 1 выдвигаемых разрядов. Одновременно со сдвигом выполняется контрольная операция, состоящая в том, что состояние контрольного разряда меняется на противоположное при нечетности суммы 1 выдвигаемых разрядов. Тогда при правильном выполнении сдвига общая четность суммы 1 в регистре после сдвига не меняется.

Операция взятия обратного кода может быть также проконтролирована путем использования кодов с проверкой четности. Если число информационных разрядов в слове четно, то число 1 в слове четно при четном 0 и число 1 нечетно при нечетном числе 0. В этом случае после образования обратного кода четность числа 1 в слове сохранится, и правильность выполнения операции можно определить, проверив сохранение четности или нечетности суммы 1 в слове, включая контрольный разряд.

Если число информационных разрядов в слове нечетно, то четному числу 1 в слове соответствует нечетное число 0, а нечетному числу 1 — четное число 0. В этом случае после образования обратного кода четность числа 1 изменится на обратную, и для проверки правильности выполнения операции необходимо взять обратный код от содержимого контрольного разряда и проверить сохранение четности или нечетности суммы 1 в слове, включая контрольный разряд.

Рассмотрим метод контроля операции сложения, использующий проверку четности.

При сложении чисел  $a$  и  $b$  разряды суммы  $S$  образуются в соответствии с выражениями

$$\begin{aligned} S_1 &= a_1 \oplus b_1 \oplus P_1; \\ S_2 &= a_2 \oplus b_2 \oplus P_2; \\ S_n &= a_n \oplus b_n \oplus P_n, \end{aligned} \quad (12.6)$$

где  $S_i$ ,  $a_i$ ,  $b_i$ ,  $P_i$  ( $i=1, 2, \dots, n$ ) — соответственно значения разрядов суммы, слагаемых и переноса, поступающего в  $i$ -й разряд. Знак  $\oplus$  означает сложение по модулю 2;  $n$  — число разрядов слагаемых и суммы.

Сложив все  $n$  приведенных выше равенств по модулю 2, получим

$$\begin{aligned} S_1 \oplus S_2 \oplus \dots \oplus S_n &= (a_1 \oplus a_2 \oplus \dots \oplus a_n) \oplus \\ &\oplus (b_1 \oplus b_2 \oplus \dots \oplus b_n) \oplus (P_1 \oplus P_2 \oplus \dots \oplus P_n). \end{aligned} \quad (12.7)$$

Поскольку сумма по модулю 2 всех разрядов слова выражает четность суммы 1 слова, последнее уравнение можно переписать в виде

$$\text{Четность } S = \text{Четность } a \oplus \text{Четность } b \oplus \text{Четность } P. \quad (12.8)$$

Таким образом, при правильном образовании суммы четность суммы ее единиц должна совпадать с четностью, определяемой выражением (12.8). Однако, как указывалось ранее, при контроле по четности не обнаруживается четное число ошибок. Сбой в схеме образования цифры разряда суммы дает единичную ошибку, и она будет обнаружена. Если сбой произойдет в схеме формирования переноса, то он может привести к распространению ошибки по многим разрядам суммы. В связи с этим для полноты контроля необходимо проверять правильность образования переноса  $P_i$ . Такой контроль может быть организован с помощью схемы, которая проверяет, что в каждом разряде существует либо перенос в прямом коде, либо инверсия переноса и не существует одновременно и то, и другое.

Другой способ заключается в дублировании схем формиро-



вания переносов и сравнения переносов основной и дублирующей схем.

Упрощенная схема контроля сумматора приведена на рис. 12.2. Для переносов и отдельно для суммы формируются контрольные разряды четности. Затем схема проверки четности проверяет выполнение условия (12.8).

Контроль выполнения арифметических операций (сложения, вычитания, умножения) можно осуществлять с помощью контрольных кодов, представляющих собой остатки от деления чисел на некоторый модуль  $R$  (контроль по модулю  $R$ ).

Если в качестве контрольного кода используется остаток по модулю  $R$ , то в качестве контрольной операции над остатками может быть выбрана та же арифметическая операция, которая производится над числами. Это вытекает из того, что для сложения, вычитания и умножения действительно соотношение

$$R(A * B) = R[R(A) * R(B)], \quad (12.9)$$

где  $R(X)$  обозначает остаток числа  $X$  по модулю  $R$ ;  $*$  — знак арифметической операции сложения, вычитания или умножения.

Контроль арифметических операций по модулю организуется следующим образом. Каждому числу, участвующему в арифме-

тической операции, ставится в соответствие контрольный код — остаток по модулю  $R$ . Одновременно с выполнением основной операции над числами та же операция производится над их контрольными кодами, и контрольный код результата основной операции сравнивается с результатом операции над контрольными кодами исходных чисел. При несовпадении фиксируется ошибка.

Чем меньше  $R$ , тем меньше разрядность конт-

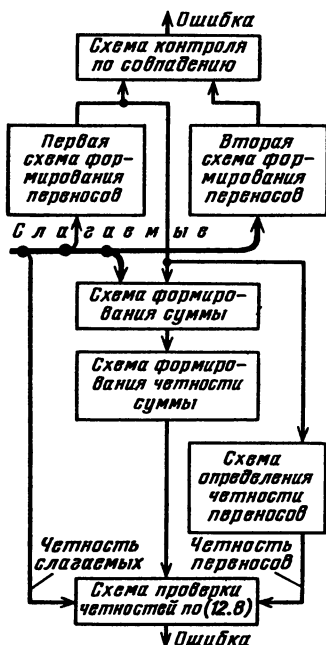


Рис. 12.2. Контроль сложения с использованием проверок по четности

рольного кода и проще дополнительная аппаратура. Для двоичных чисел контроль по модулю возможен при  $R \geq 3$ , поэтому в ЭВМ часто используют контроль по модулю 3.

Покажем, что с помощью контроля по модулю 3 обнаруживаются любые одиночные ошибки. Одиночная ошибка в каком-либо разряде двоичного числа соответствует изменению числа на значение  $\pm 2^i$ . Для обнаружения ошибки необходимо, чтобы контрольные коды чисел  $a$  и  $a \pm 2^i$  не совпадали, т. е.

$$R(a) \neq R(a \pm 2^i) = R(a) \pm R(2^i),$$

или

$$R(2^i) \neq 0.$$

Но  $2^i$  не делится на 3 без остатка, следовательно, требуемое условие выполняется. Кроме одиночных ошибок при контроле по модулю 3 обнаруживается часть двойных ошибок, а именно те ошибки, при которых правильные и ошибочные результаты имеют несовпадающие остатки от деления на 3.

На рис. 12.3 изображена структурная схема сумматора с контролем по модулю  $R$ . Отметим, что контрольный блок значительно проще основного, при  $R=3$  он содержит только два двоичных разряда. При реализации контроля важное значение имеет построение схем формирования остатков при минимальных затратах оборудования. Очевидно, что нахождение остатка путем прямого деления двоичного числа на 3 — путь неприемлемый. Однако кодирование по модулю 3 обладает свойствами, позволяющими строить достаточно простые комбинационные схемы формирования остатка по модулю 3.

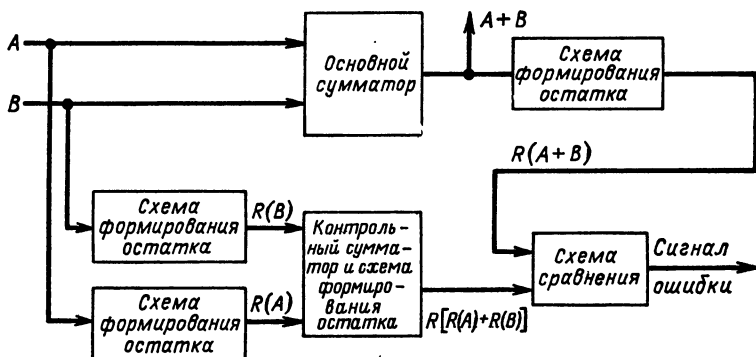


Рис. 12.3. Структурная схема сумматора с контролем по модулю  $R$

Может быть применен контроль по модулю с большим основанием, чем 3, например по модулю 7. При этом увеличивается число кратных ошибок, которые могут обнаруживаться системой контроля, однако возрастает сложность кодирующей и декодирующей аппаратуры.

Повышение степени интеграции и снижение стоимости (в пересчете на один клапан) электронных схем сделали возможным практическое использование для контроля некоторых узлов, например сумматоров, самопроверяемого дублирования их схем, при котором ошибки обнаруживаются по несовпадению сигналов на выходах дублированных схем. Этим способом обнаруживают ошибки любой кратности.

#### **12.4. Взаимодействие систем автоматического контроля, восстановления вычислительного процесса и диагностирования**

В ЭВМ, снабженной системой автоматического контроля, возникновение ошибок в каком-либо устройстве порождает сигнал ошибки, с появлением которого приостанавливается выполнение программы целиком или только рабочей процедуры в неисправном устройстве. При этом 1 в соответствующем разряде регистра ошибок, высвечиваемом на сигнальном табло на пульте оператора, указывает укрупненно место, где обнаружена ошибка (устройство, узел, регистр, группа разрядов регистра и т. д.). Сигнал ошибки инициирует работу системы восстановления.

Система автоматического восстановления во взаимодействии с системой автоматического контроля обычно выполняет следующие функции:

1) распознавание характера обнаруженной ошибки, т. е. выяснение, вызвана ошибка случайным сбоем, перемежающимся или устойчивым отказом;

2) организация «рестарта», т. е. продолжения выполнения программы путем устранения возникающей ошибки в информации повторением ошибочно выполненной микрооперации, команды или сегмента программы (при обнаружении, что ошибка вызвана сбоем);

3) запись в память информации о сбое;

4) инициирование (при обнаружении отказа) работы системы автоматического диагностирования (САД).

На САД в данном случае возлагается:

1) локализация места отказа с заданной степенью подробности, например до уровня сменной платы, и, если возможно, реконфигурация, т. е. отключение неисправного узла или

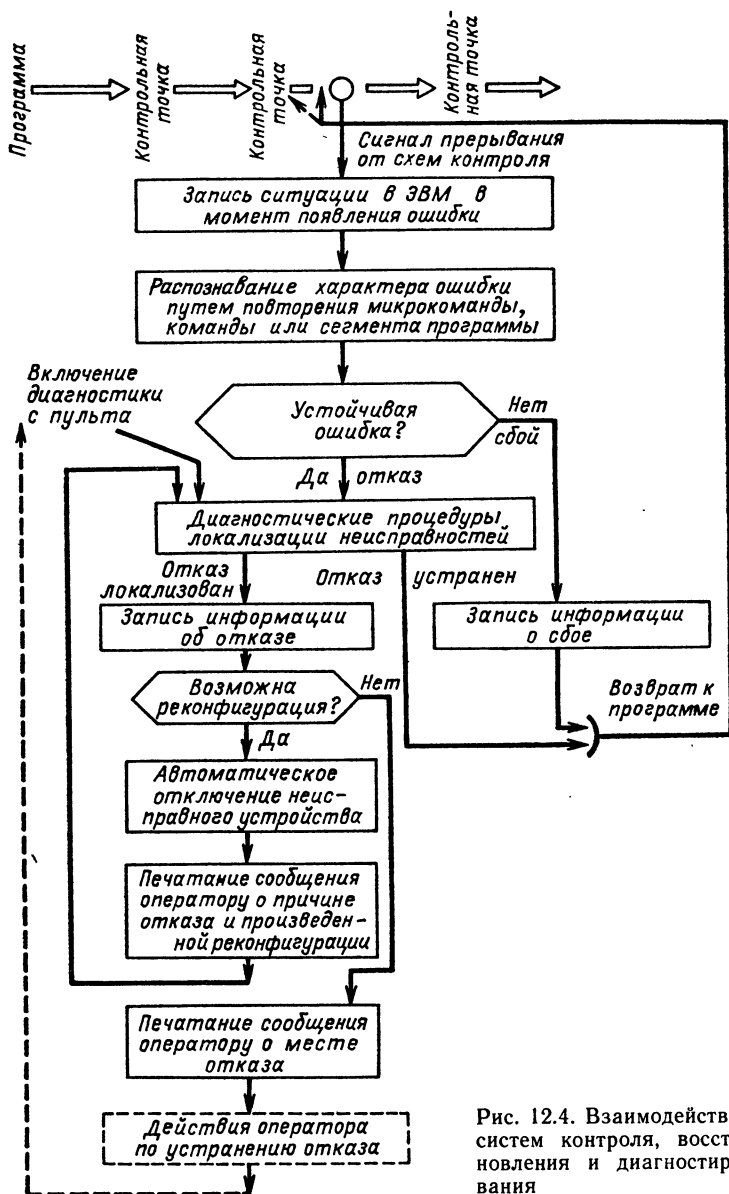


Рис. 12.4. Взаимодействие систем контроля, восстановления и диагностирования

устройства с передачей его функции другому соответствующему узлу или устройству;

2) запись в память информации об отказе для последующей обработки.

Общая логика взаимодействия систем автоматического контроля, восстановления и диагностирования показана на рис. 12.4. Первой выполняется процедура обработки сигнала прерывания от системы контроля, начинающаяся записью состояний регистров процессора и старого слова состояния программы (ССП) в соответствующие ячейки памяти и выборкой нового ССП системы восстановления. Далее производится запись в специальные регистры или ячейки памяти ситуации в ЭВМ в момент обнаружения ошибки (записывается состояние запоминающих элементов и шин передачи данных неисправного устройства) <sup>1</sup>.

Для распознавания характера ошибки (сбой или отказ) необходимо повторить, может быть, даже несколько раз, операцию, в которой обнаружилась ошибка. Однако желательно это сделать с минимальными потерями времени. С этой точки зрения лучше всего, если процессор и система контроля выполнены таким образом, что проверяется правильность каждой микрооперации и вычислительный процесс останавливается на ошибочно выполненной микрооперации, которая может быть повторена. Микрооперация (команда) может быть повторена, если не искажалась используемая в операции информация (операнды, адреса и т. д.), т. е. если не пройден «порог повторения». Это проверяет входящая в систему программа обработки ошибок, которая по записанной ситуации, соответствующей появлению ошибки, определяет, пройден или нет порог повторения микрооперации или команды <sup>2</sup>. Если соответствующий порог пройден, то вместо микрооперации повторяется команда, а вместо команды — сегмент программы. Последнее возможно, если программист предусмотрел в программе «контрольные точки», сохраняющие промежуточные данные, позволяющие повторить программу с данного места.

Если при повторении микрокоманды (команды, сегмента, программы) ошибка не повторяется, событие распознается как сбой и происходит возврат к точке прерывания программы, но перед этим в память записывается дополнительная информация

---

<sup>1</sup> В случае отказа периферийного устройства используется информация о его состоянии, доставляемая командой «Уточнить состояние».

<sup>2</sup> Операции ввода-вывода допускают повторения на уровне команды.

о состоянии в момент сбоя вычислительной системы (какие в это время выполнялись операции в периферийных и других устройствах, название программы, адрес команды, операнды, время). Информация о ситуациях при сбоях и об отказах накапливается во внешнем ЗУ и в последующем обрабатывается специальной программой, вырабатывающей определенные рекомендации обслуживающему персоналу, выполняющему профилактические работы.

Если при определенном числе повторений (например, восемь) ошибка сохраняется, событие распознается как отказ и автоматически приводится в действие САД для определения места неисправности. Диагностические процедуры также могут включаться вручную с пульта, например при пуске машины.

После выявления места отказа проверяется возможность реконфигурации путем автоматического отключения неисправного устройства и передачи его функций другому устройству. Если это возможно, то производится реконфигурация системы. Затем после сообщения оператору об отказе и произведенной реконфигурации происходит возврат к контрольной точке программы. Если реконфигурация невозможна, САД инициирует на пульте код неисправности. Оператор, пользуясь справочником неисправностей, находит неисправный блок, ремонтирует или заменяет его, затем с пульта включает диагностические процедуры. Если отказ устранен, производится возврат к контрольной точке программы.

## **12.5. Принципы построения систем автоматического диагностирования ЭВМ**

В техническом обслуживании ЭВМ наиболее сложной проблемой является поиск места (локализация) неисправности, вызвавшей отказ в работе машины. Поиск места неисправности вручную в таком сложном объекте, как современные ЭВМ, требует очень высокой квалификации обслуживающего персонала и больших затрат времени.

Для облегчения и ускорения поиска причины отказа ЭВМ снабжаются *системами автоматического диагностирования неисправностей*.

Диагностирование некоторого объекта может производиться функциональным или тестовым способом. Функциональное диагностирование предполагает контроль при помощи тех или иных средств контроля правильности функционирования объекта диагностирования при реализации его рабочего процесса. Тестовое диагностирование производится путем подачи на ди-

агностируемый объект специальных тестовых воздействий и сравнения реакций объекта на эти воздействия с эталонными ответами.

Применяемые в ЭВМ средства автоматического контроля правильности функционирования помимо главной своей задачи — обнаружение ошибки в работе машины — позволяют диагностировать (локализовать) место неисправности, но с точностью лишь до устройства или крупного блока (например, модуля ОП, АЛУ и т. п.), т. е. с разрешающей способностью, которая для большинства ЭВМ оказывается недостаточной.

В ЭВМ используется главным образом метод тестового диагностирования, позволяющий локализовать место неисправности с достаточной разрешающей способностью.

Диагностические процедуры могут осуществляться путем пропуска на ЭВМ специальных диагностических программ. Однако несравненно более гибкое диагностирование с большей детализацией места неисправности (большей разрешающей способностью) обеспечивают диагностические процедуры, выполняемые под управлением специальных диагностических микротестов и микропрограмм (*«микропрограммное диагностирование»*, или, короче, *«микродиагностика»*).

В связи со сложностью аппаратуры диагностирование ЭВМ производится в форме многоэтапного процесса, причем на разных этапах используются различные средства САД — микропрограммные, программные, а также некоторые аппаратурные средства системы диагностирования.

Система автоматического диагностирования представляет собой комплекс аппаратурных, микропрограммных и программных средств и справочной документации (справочников неисправностей, инструкций, схем ЭВМ, тестов).

В зависимости от размещения аппаратурных средств системы диагностирования различают встроенные САД, когда диагностирующие средства размещаются внутри ЭВМ, и внешние САД, когда эти средства находятся вне машины. На практике часто САД строятся комбинированными: одна часть их средств встраивается в машину, а другая оформляется в виде специального дополнительного оборудования, подсоединяемого к ЭВМ при диагностировании неисправностей.

Обычно объем диагностических микропрограмм (тестов «микродиагностики») и программ столь велик, что их не удастся хранить внутри ЭВМ (в УП, ОП или специальных встроенных памятьях), но это в ряде случаев и не так важно, так как диагностические тесты не так часто выполняются. Поэтому обычно они хранятся во внешних ЗУ, в том числе в специальных внешних ЗУ САД.

Для САД современных средних и больших ЭВМ общего назначения характерным является наличие в их составе построенных на основе микропроцессоров или микро-ЭВМ специализированных диагностических процессоров (называемых *сервисными адаптерами* или *сервисными процессорами*), управляющих загрузкой в ЭВМ из внешних ЗУ («сервисных ЗУ») диагностической информации, иницированием диагностических процедур, опросом состояния ЭВМ после подачи тестовых воздействий, сравнением полученных ответов с эталонными, индикацией и регистрацией результатов диагностирования.

При построении САД ЭВМ широкого назначения (например, в ЕС ЭВМ) используется диагностирование по методу «раскрутки». Метод «раскрутки» предполагает поэтапное последовательное расширение работоспособной части ЭВМ путем включения в эту часть оборудования, проверенного на предыдущем этапе.

На каждом  $i$ -м этапе диагностического процесса подмашина  $M_i$ , представляющая собой часть оборудования проверяемой ЭВМ, участвует в диагностировании аппаратуры  $\Delta M_i$ , которая, если в ней нет отказов, присоединяется к подмашине  $M_i$ , образуя новую подмашину  $M_{i+1} = M_i \cup \Delta M_i$ , выполняющую диагностическую процедуру на следующем этапе.

Подмашина  $M_0$ , с которой начинается «раскрутка», называется *диагностическим ядром* ЭВМ. Диагностическое ядро должно иметь повышенную надежность и допускать проверку работоспособности вручную или в полуавтоматическом режиме.

*Системы автоматического диагностирования в машинах ЕС ЭВМ.* В рассматриваемых САД реализовано микропрограммное диагностирование (микродиагностика).

Микропрограммное диагностирование в отличие от диагностирования программными средствами позволяет контролировать изменение состояния аппаратуры ЭВМ на каждом машинном такте, благодаря чему достигается высокая разрешающая способность САД — один-два сменных элемента (ТЭЭ). Однако при этом требуется дополнительная аппаратура (около 3—5 % объема оборудования процессора и каналов) и необходима разработка большого объема (около 1 Мбайт) микродиагностической информации (микротестов).

Средства САД и, в частности, микродиагностирование в некоторых моделях ЭВМ используются системой автоматизации профилактических испытаний, выполняемых при программно-управляемом изменении уровней питающих напряжений на  $\pm 5$  % номинальных значений.

Системы диагностирования ЭВМ ЕС используют дополнительное внешнее оборудование — сервисный адаптер (сервисный процессор) с собственной небольшой оперативной памятью



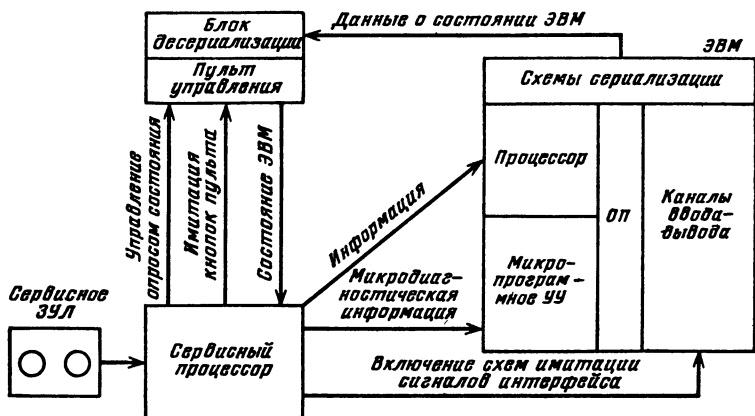


Рис. 12.5. Система автоматического диагностирования ЭВМ ЕС-1045

и сервисное кассетное ЗУЛ (*пульт*овый накопитель). На рис. 12.5 показаны связи этих сервисных устройств с оборудованием ЭВМ в САД ЭВМ ЕС-1045. Микродиагностическая информация, а в ряде моделей ЕС ЭВМ и рабочие микропрограммы машины хранятся на кассетах сервисного ЗУЛ.

Управляющая память (УП) микропрограммного управляющего устройства процессора машины вся или частично является загружаемой.

Из сервисного ЗУЛ передаются в регистр микрокоманд УУ процессора микротесты, в загружаемую УП (или ее загружаемую секцию) — диагностические микропрограммы. Кроме того, из сервисного ЗУЛ производится также загрузка диагностических программ в ОП ЭВМ.

Сервисный процессор имеет собственную ориентированную на диагностические операции систему команд. Эти команды поступают в сервисный процессор с сервисного ЗУЛ и дешифрируются дешифратором диагностических операций сервисного процессора.

Данные о текущем состоянии машины, т. е. данные о состоянии ее регистров, счетчиков, управляющих триггеров, уровней сигналов на шинах (общий объем этих данных в ЭВМ ЕС составляет 3—4 тыс. бит), должны передаваться на пульт управления машины, а в режиме диагностирования — выборочно и в сервисный процессор для сравнения реакций аппаратуры на микротесты и диагностические микропрограммы с эталонными, содержащимися в микродиагностической информации.

Передача из машины на пульт управления и в сервисный

процессор указанного объема данных параллельным кодом практически невозможна, так как требует нескольких тысяч проводов. Поэтому эти данные подвергаются *сериализации*, другими словами, преобразованию объединенного параллельного кода состояния ЭВМ в последовательный. Этот последовательный код передается на пульт управления машины, там преобразуется в поток байт (*десериализация*) и отображается на индикаторах пульта, запоминается в памяти пульта, а нужные фрагменты данных о состоянии ЭВМ, адресуемые сервисным процессором, поступают в последний для сравнения с эталонными кодами. Сериализация данных о состоянии ЭВМ производится многоступенчатой схемой, построенной на мультиплексорах (см. гл. 3) [27].

В ЕС ЭВМ диагностирование машин производится по методу «раскрутки». Диагностическое ядро образуют сервисный процессор, сервисное ЗУЛ и часть аппаратуры пульта управления.

Диагностирование производится в несколько этапов. Можно выделить следующие укрупненные этапы: 1) статическое микродиагностирование; 2) динамическое микродиагностирование; 3) функциональное (программное) диагностирование.

*Статическое микродиагностирование.* Управление процессором машины берет на себя сервисный процессор. Производится загрузка из сервисного ЗУЛ регистра микрокоманд микротестом и иницируется выполнением микрооперации с заданным тестовым набором.

Одновременно из сервисного ЗУЛ поступают эталонный код и адрес (номер) байта в сериализованном потоке данных о состоянии ЭВМ, который следует сравнить с эталоном.

В данном случае микродиагностирование называется «статическим», так как проверяемая аппаратура работает не с рабочей тактовой частотой, а в темпе, определяемом загрузкой оборудования микротестами. На данном этапе проверяются аппаратура процессора, каналов, адаптера канал — канал, управление памятью, загружаемая УП, частично другие памяти.

*Динамическое микродиагностирование* производится при работе аппаратуры ЭВМ с рабочей тактовой частотой, что позволяет выявить неисправности, связанные с искажениями в реализации временных диаграмм. При этом используется проверенная на предыдущем этапе загружаемая УП, в которую с сервисного ЗУЛ загружаются диагностические микропрограммы, принимающие на себя управление ЭВМ. Сервисный процессор сравнивает результаты выполнения микропрограмм с эталонами. На этом этапе в динамическом режиме проверяется аппаратура ЭВМ, в том числе все ее памяти.

*Функциональное (программное) диагностирование.* Сред-

ствами функционального диагностирования служат проверенная на предыдущих этапах аппаратура машины и диагностические программы: 1) программа «базовый тест»; 2) программы тест-секций, выполняемые под управлением мониторной программы (*диагностического монитора*); 3) программы автономных диагностических тестов отдельных устройств. Базовый тест, загружаемый с сервисного ЗУЛ, проверяет выполнение группы команд процессора («ядро системы команд»), в том числе команд ввода-вывода, цепи связи с некоторыми периферийными устройствами.

Далее при помощи программ тест-секций производится диагностирование всех операций процессора, системы прерывания, защиты памяти, таймера, средств прямого управления, ОП, памяти ключей защиты, каналов ввода-вывода, периферийных устройств и их УПУ.

Если результат прохождения какого-либо теста оказывается неудачным, САД высвечивает на индикаторном табло пульта управления номер этого теста. Оператор, пользуясь справочником неисправностей, по номеру теста находит место неисправности. В такой форме система сообщает оператору о месте неисправности на всех этапах — от начала диагностического процесса до этапа проверки мультиплексного канала включительно. В последующих за этим этапах диагностики система выдает на печатающее устройство сообщение оператору о месте неисправности и указание, какая сменная плата (ТЭЗ) подлежит замене.

Более подробно вопросы автоматизации контроля и диагностирования ЭВМ рассмотрены в [27]. Там же изложены принципы построения таких средств повышения обслуживаемости ЭВМ, как системы автоматизации профилактических испытаний, сбора и обработки данных об отказах и сбоях, дистанционного обслуживания ЭВМ, а также особенности средств поддержки эксплуатационного обслуживания микропроцессорных устройств и персональных компьютеров.

### **Контрольные вопросы**

1. Перечислить основные характеристики надежности ЭВМ. Какие из этих характеристик являются специфичными для ЭВМ как универсального преобразователя информации? В чем различие между отказом и сбоем в работе ЭВМ?

2. Обоснуйте, исходя из выражения (12.1), необходимый состав аппаратурно-программных средств, поддерживающих техническое обслуживание ЭВМ и способствующих повышению коэффициента использования ЭВМ и снижению расходов на эксплуатацию машины.

3. Какой кратности ошибки (кроме одиночных) обнаруживает контроль по четности (нечетности)?

4. Сколько нужно дополнительных (контрольных) разрядов для реализации памяти с коррекцией одной ошибки с помощью кода Хэмминга при длине информационного слова 16, 24, 32 разряда?

5. Как в процессе взаимодействия систем контроля, восстановления вычислительного процесса и диагностирования определяется, является ошибка машины результатом отказа или сбоя? В чем различие функционирования этих систем в случае сбоя и отказа?

## *Глава 13*

# **ПРИНЦИПЫ ОРГАНИЗАЦИИ МУЛЬТИПРОГРАММНЫХ ЭВМ ОБЩЕГО НАЗНАЧЕНИЯ**

## **13.1. Предварительные замечания**

В Советском Союзе ЭВМ общего назначения представлены имеющими единую архитектуру программно-совместными машинами, образующими Единую систему ЭВМ (ЕС ЭВМ).

Общее назначение машин состоит в том, что они обладают универсальными сбалансированными по показателям эффективности и производительности возможностями:

- по решению разнообразных задач: научно-технических, информационно-логических, планово-экономических, управленческих, учебно-статистических;

- по сбору, накоплению, хранению и обработке больших объемов разнообразной информации;

- по представлению в машине, обработке и редактированию разнообразных видов и форматов данных, а также по выполнению с высокой скоростью вычислений с различной (в том числе и очень высокой) точностью;

- по использованию в составе ЭВМ разнообразных по количественным и качественным показателям наборов периферийных устройств с обеспечением высокой интенсивности обмена информацией между ядром ЭВМ и периферийным оборудованием;

- по реализации разнообразных высокоэффективных режимов работы и общения пользователей с ЭВМ, в том числе мультипрограммного, многозадачного, диалогового, разделения времени, многопользовательского (коллективного пользования), реального времени и др.;

- по использованию огромного объема разнообразных прикладных и системных программных продуктов.

Однако за универсальность приходится расплачиваться сложностью логической организации, аппаратурных и системных программных средств. Современные ЭВМ общего назначения — одни из наиболее сложных объектов вычислительной техники.

ЭВМ общего назначения со сбалансированными характеристиками для решения как научно-технических, так и разного рода информационных и планово-экономических задач являются в настоящее время вычислительными средствами, выполняющими в стране большой объем работ по машинной обработке информации и вычислениям в сферах науки, сложных инженерных задач и автоматизированных информационно-управляющих систем. Во многих случаях оказывается целесообразным комплексирование ЭВМ общего назначения с несколькими (иногда десятками и сотнями) персональными компьютерами, микро- и мини-ЭВМ с распределением между ними и ЭВМ общего назначения функций в соответствии с их потенциальными возможностями по обработке и хранению информации и предоставляемыми пользователю удобствами общения.

Сказанное объясняет, почему изучение особенностей организации ЭВМ общего назначения занимает в практическом и познавательном смысле важное место при подготовке специалистов в области создания и использования вычислительной техники.

Многие вопросы архитектуры этих машин, построения и функционирования их устройств рассмотрены в предыдущих главах. В гл. 13 и 14 продолжено рассмотрение вопросов организации ЭВМ общего назначения, причем основное внимание уделено особенностям режимов работы этих машин и средствам поддержки этих режимов. В § 14.7 приведены соображения об основных тенденциях развития ЭВМ общего назначения в ближайшие годы.

### **13.2. Организация мультипрограммного режима работы ЭВМ**

Важным вопросом в организации вычислительного процесса в ЭВМ общего назначения является использование мультипрограммирования, которое предполагает, грубо говоря, одновременное выполнение ЭВМ нескольких программ. Естественно, в каждый момент времени ЭВМ выполняет команду какой-то одной программы. Однако всякий раз, когда выполнение процессором некоторой программы приостанавливается из-за необходимости произвести, например, операцию ввода-вывода, процессор переходит к обработке другой готовой для выполнения программы. При этом предполагается, что одновременно в оперативной памяти присутствует несколько программ (или их фрагментов),

которые могут находиться в состояниях *активном* (программа обрабатывается на процессоре), *готовности* к обработке или *ожидания* некоторого события, например завершения операции ввода-вывода или освобождения нужного ресурса.

Программа находится в состоянии готовности, если ей предоставлены все необходимые ресурсы, кроме времени процессора.

Мультипрограммирование предназначено для повышения пропускной способности вычислительной установки путем более равномерной и плотной загрузки всего ее оборудования, в первую очередь процессора. При этом скорость работы самого процессора и номинальная производительность ЭВМ, как она определена в гл. 1, не зависят от мультипрограммирования. Существенной характеристикой для пользователя является пропускная способность, которая оценивается средним объемом вычислений, выполняемых ВС в единицу времени при решении наборов практических задач.

На рис. 13.1 показано выполнение двух программ при однопрограммном и мультипрограммном режимах работы. Как видно из рисунка, общее время выполнения программ *A* и *B* при мультипрограммном режиме значительно меньше, чем при однопрограммном. В рассмотренном примере в мультипрограммном режиме сохранились (хотя и существенно уменьшились) паузы в работе процессора. Дальнейшее увеличение пропускной способности в рассматриваемом примере можно получить, увеличив число одновременно обрабатываемых программ (задач) (*коэффициент мультипрограммности*).

Поскольку участвующие в мультипрограммной обработке программы конкурируют за получение времени процессора, доступа к памяти и другим устройствам, между ними должны быть установлены приоритетные соотношения.

Заметим, что переключение программ из состояния готовно-

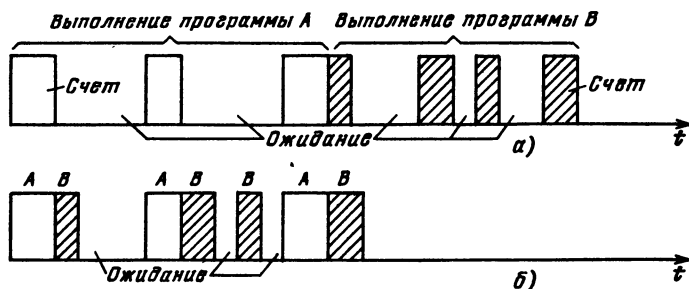


Рис. 13.1. Выполнение программ в однопрограммном (а) и мультипрограммном (б) режимах

сти в активное состояние связано с некоторыми издержками времени на работу управляющей программы.

Практически за счет мультипрограммирования удается повысить пропускную способность, а следовательно, загрузку процессора в 4—5 раз.

Из сказанного следует, что мультипрограммирование является также средством уменьшения времени выполнения заданного набора программ. Более того, если обрабатывается всего одна программа, то за счет распараллеливания программы и мультипрограммной обработки удается сократить время ее выполнения.

Распараллеливание программ является широко используемым приемом повышения коэффициента мультипрограммности.

Управляющая программа операционной системы, реализуя мультипрограммный режим, должна распределять (в том числе и динамически) между параллельно выполняемыми программами и участками программы ресурсы системы (время процессора, основную и внешнюю памяти, каналы, устройства ввода-вывода, ключи защиты памяти и др.), с тем чтобы достигалось увеличение пропускной способности с учетом ограничений на ресурсы и требований по срочности выполнения отдельных программ.

Для пользователя внешней единицы работы ВС служит *задание*, соответствующее некоторой совокупности вычислений и представляющее для него законченную работу (расчет сменного задания цехам предприятия, расчет потребности в материалах и т. д.).

Задания не зависят друг от друга и поэтому могут группироваться в пакеты и выполняться в мультипрограммном режиме без синхронизации завершения обработки одних заданий с началом обработки других.

Задание разбивается на выполняемые последовательно части, называемые пунктами задания. Пункту задания соответствует внутренняя единица работы — задача, которую можно рассматривать как совокупность ее программы, используемых наборов данных, необходимых ей ресурсов.

#### *Организация обработки пакета заданий в мультипрограммном режиме*

Организация обработки пакета заданий показана на рис. 13.2. Предварительно на языке управления заданиями составляется описание задания и его пунктов, в котором для задания указываются необходимая емкость ОП, приоритет и класс, а для каждого пункта — имена используемых программ, входных и выходных наборов данных, приоритет пункта, не-

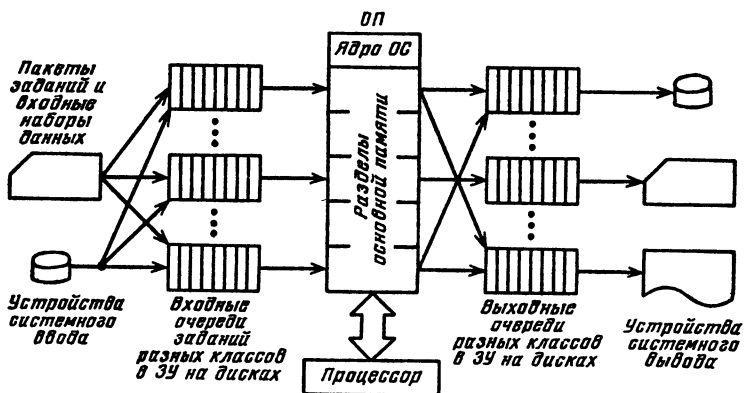


Рис. 13.2. Организация обработки пакета заданий в мультипрограммном режиме

обходимые емкость ОП и устройства. Описания заданий объединяются в пакет.

Обработка пакета заданий организуется и управляется программами операционной системы (программой супервизор и др.).

Тексты заданий и входные наборы данных вводятся *программой системного ввода с устройств системного ввода* (перфокартный ввод, ЗУ на лентах или дисках). Очереди заданий, упорядоченные для каждого класса в соответствии с приоритетом заданий, располагаются в ЗУ прямого доступа (ЗУ на дисках).

Программа *Планировщик* выбирает задания из очереди старшего класса, назначает ему периферийные устройства, выдает сообщение оператору об установке нужных томов (пакетов дисков, катушек с лентой) и загружает программу очередного пункта задания.

Средства супервизора управляют обработкой задач в мультипрограммном режиме, распараллеливают процесс обработки программы, распределяют ресурсы (память, время процессора и др.) между выполняемыми параллельно задачами, следят за освобождением задачами ресурсов, производят динамическую загрузку программ, обрабатывают возникающие прерывания.

Задача, находящаяся в состоянии готовности, переводится супервизором в активное состояние и начинает обрабатываться процессором, если все другие совместно выполняемые задачи с более высоким приоритетом оказываются в состоянии ожидания. Обработка этой задачи будет прервана супервизором, если перейдет в состояние готовности какая-либо задача более высокого приоритета.



Группам заданий (задач), находящимся в ОП, может быть назначена обработка с *квантованием времени*, при которой супервизор поочередно выделяет заданиям один и тот же интервал (*квант*) времени работы процессора. Квантованная обработка позволяет избежать монополизации процессора одной задачей, что особенно важно при диалоговом режиме работы ВС.

Результаты выполнения задания (задачи) помещаются в ЗУ прямого доступа (ЗУ на дисках) в выходную очередь, соответствующую выходному классу задания.

Выходные данные выводятся *программой системного вывода* на одно из *устройств системного вывода* (перфокартный вывод, печатающее устройство, ЗУ на ленте).

Для организации мультипрограммного вычислительного процесса ВС требуются определенные аппаратурные и программные средства.

*Супервизор* — программа, осуществляющая планирование вычислительного процесса, распределение ресурсов системы, обработку прерываний.

*Прерывание программ.* Часть функций, которые выполняет система прерывания мультипрограммной ЭВМ, свойственна и однопрограммным системам: обеспечение параллельной работы периферийных устройств и процессора, синхронизация работы ЭВМ с внешними по отношению к ней устройствами. Некоторые другие функции являются специфическими для мультипрограммного режима. Если, например, в однопрограммной машине произошло обращение к памяти по несуществующему в данной конфигурации адресу, то машина остановится. При мультипрограммной работе в подобных ситуациях останавливается выполнение только данной программы, но не система. Следовательно, каждая ситуация, подобная указанной выше, должна вызывать запрос прерывания, позволяющий супервизору указать пользователю на его ошибку, не прекращая исполнения других программ. Имеются и другие особенности прерывания в мультипрограммных системах.

Часть функций системы может выполняться только под управлением супервизора. Целевая программа должна для этого передать управление супервизору. В систему команд мультипрограммной системы должна быть включена команда «Вызов супервизора», которая производит прерывание целевой программы и снабжает супервизор информацией о характере затребованного действия.

В однопрограммных машинах маскирование какого-либо запроса прерывания обычно приводит к тому, что он запоминается до тех пор, пока запрещение прерывания не будет снято. В мультипрограммных системах действие маски прерывания должно

иметь избирательный характер. Прерывания, которые должны воздействовать только на какую-либо одну целевую программу, при маскировании должны игнорироваться. В противном случае они могут нарушить работу другой целевой программы, если в последней маска для данного прерывания снята.

*Динамическое распределение памяти* осуществляется между программами в ходе самого вычислительного процесса.

*Защита памяти* предотвращает непредусмотренное воздействие одной программы на другую путем запрета вторжения одних программ в области памяти других программ. При отсутствии защиты памяти ошибки в одних программах могут приводить к порче информации, принадлежащей другим программам.

*Привилегированные команды.* В мультипрограммной системе часть команд может выполняться только супервизором. Такие команды называют привилегированными.

К привилегированным операциям относятся установка и изменение кодов защиты памяти, установка и изменение маски прерывания, останов процессора, команды управления устройствами ввода-вывода и др. В мультипрограммных системах в состав процессора вводят триггер, состояние которого задает режим либо *супервизора*, либо целевой программы (*задачи*). Появление привилегированной команды в режиме *задачи* вызывает прерывание.

*Датчик времени (таймер).* Переключение программ из активного режима в пассивный в мультипрограммной системе с квантованной обработкой производится не только при достижении программой такого состояния, когда она не может дальше работать (например, ожидание ввода-вывода), но и по истечении определенного промежутка времени, оговоренного данной программой под цикл активности. Кроме того, должен вестись учет времени, фактически использованного каждой целевой программой для последующей оплаты машинного времени пользователем. Датчик времени представляет собой схему, позволяющую измерять интервал времени между различными программными событиями.

### **13.3. Структура и принцип действия ВС коллективного пользования (разделения времени)**

Эффективность вычислительной техники в очень большой степени определяется производительностью труда лиц, решающих задачи на машинах. Эта производительность во многом зависит от предоставляемых средствами вычислительной системы возможностей взаимодействия и общения пользователей с ма-

шиной в процессе подготовки и отладки программ, а также в процессе решения задач.

При переходе к ЭВМ второго поколения для повышения эффективности использования собственно средств вычислительных систем стал применяться операторный (пакетный) режим решения задач на ЭВМ, при котором пользователи сдают свои задачи оператору ЭВМ, формирующему пакет задач, решаемых, как правило, в мультипрограммном режиме.

В результате пользователь утратил непосредственную связь с ЭВМ, характерную для эксплуатации машин первого поколения.

Ожидание результатов решения по переданной оператору программе затрудняет отладку программ, делает невозможным оперативное внесение в программы изменений по результатам решения, что ведет к снижению эффективности труда программистов по сравнению с эффективностью в режиме непосредственного общения с ЭВМ.

Выход из создавшейся ситуации был найден в идее вычислительной системы коллективного пользования (ВСКП), в которой реализуется практически одновременный доступ многих независимых пользователей к ресурсам ВС (многопользовательский режим). При этом предполагается наличие у пользователя своего терминала, например дисплея, с помощью которого он может независимо от других пользователей обратиться к системе с запросами на обработку его программ и получить результаты обработки. При одновременном обращении нескольких пользователей ВС должна реагировать на их запросы с задержкой, которая пользователю представляется почти такой же, как и при индивидуальном пользовании. Независимость пользователей означает, что они могут при подготовке своих программ считать, что каждый из них располагает всеми ресурсами системы, в том числе всей ее памятью.

Во многих случаях ВСКП создаются на базе ЭВМ общего назначения, чему способствуют их высокая производительность, возможность решения на них разнообразных задач, наличие в их составе обширного комплекса периферийных устройств (в том числе ЗУ большой емкости), а главное, эффективных аппаратурных и программных средств поддержки мультипрограммного режима, составляющего основу для коллективного использования ЭВМ.

Если все запросы пользователей имеют примерно одинаковую длину и не требуют много машинного времени, как это имеет место в информационно-справочных системах или автоматизированных обучающих системах, то запросы формируются в пакет и обрабатываются согласно дисциплине обслуживания «первый

пришел — первый ушел» или какой-либо другой, в том числе и с учетом различных приоритетов запросов. Такие ВСКП (система типа «запрос — ответ») работают без квантованного обслуживания запросов.

Более характерным является случай, когда в ВСКП поступают запросы, сильно отличающиеся по необходимому машинному времени, причем заранее нельзя определить продолжительность обработки того или иного запроса. Такая ситуация характерна для любого вычислительного центра без постоянной номенклатуры расчетов.

В этом случае поддержание в ВСКП у пользователей иллюзии прямого общения с ВС и обладания ее ресурсами требует уменьшения времени отклика системы в первую очередь на короткие запросы. Действительно, пользователи, затребовавшие выполнения программ, требующих много машинного времени, склонны спокойно ждать обслуживания сравнительно долго. В то же время пользователи, запрашивающие решение коротких задач, оказываются более нетерпеливыми, и поэтому такие запросы должны получить обслуживание в первую очередь.

Для создания эффекта прямого контакта пользователя с системой путем сокращения времени ожидания ответа на короткие запросы в ВСКП применяют *квантованное обслуживание*, или, другими словами, *разделение времени* при выполнении запросов.

*Вычислительная система с разделением времени (СРВ)*, работающая в мультипрограммном режиме, в определенной последовательности обслуживает пользователей, выделяя программе (запросу) каждого пользователя некоторый интервал времени (*квант обслуживания*). Если в течение выделенного программе кванта времени ее обработка не заканчивается, программа прерывается и становится в очередь программ, ожидающих обработки.

Порядок распределения между пользователями основного ресурса СРВ — времени процессора и соответственно других ее ресурсов — устанавливается дисциплиной обслуживания. Дисциплина обслуживания выбирается с учетом следующих требований: обеспечение режима прямого контакта пользователя с вычислительной системой, высокий уровень эффективности использования технических средств вычислительной системы, простота аппаратных и программных средств, обеспечивающих реализацию принятой дисциплины обслуживания. Поскольку нельзя допустить фактической монополизации времени процессора и других средств вычислительной системы каким-либо пользователем, то кажется естественной и одновременно технически просто осуществимой дисциплиной обслуживания, при которой всем

программам выделяется один и тот же квант времени, по возможности меньший, чтобы промежутки между интервалами активности программ пользователя были минимальными.

Однако уменьшение выделяемого программам кванта времени ведет к снижению эффективности использования технических средств системы, так как переключение программ из пассивной фазы в активную сопровождается служебными операциями, доля которых при этом в общем времени работы системы возрастает. Значение кванта времени выбирается в результате разумного компромисса, с тем чтобы и время ожидания ответа на запрос, и потери производительности от переключения активности программ были приемлемыми.

*Дисциплины обслуживания в СРВ.* Было предложено несколько дисциплин обслуживания пользователей в СРВ. Их можно разбить на две основные группы: одноочередные и многоочередные.

*Одноочередная дисциплина обслуживания.* Схема одноочередной СРВ представлена на рис. 13.3. Вновь поступающие от пользователей запросы (программы) ставятся в конец очереди. Для обслуживания выбирается программа из начала очереди, и ей отводится квант времени  $\Delta$  процессора. Если за это время программа успеет завершиться, результат будет выдан пользователю, а процессор приступит к обработке следующей из очереди программы. Если программа за выделенный квант времени не закончится, обработка ее прервется и она поступит в конец очереди.

Иногда в целях уменьшения времени отклика системы на короткие (по времени выполнения) запросы рассматриваемая дисциплина модифицируется, и вновь поступающий в систему

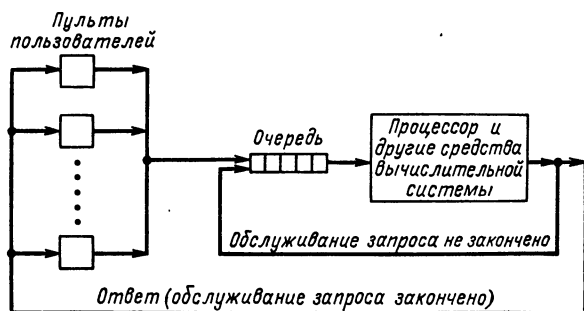
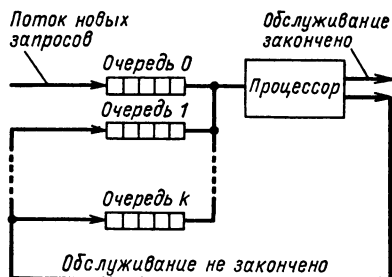


Рис. 13.3. Система разделения времени с одноочередной дисциплиной обслуживания

Рис. 13.4. Система разделения времени с многоочередной дисциплиной обслуживания



запрос становится в начало очереди. В остальном процесс функционирования остается без изменения.

**Многоочередная дисциплина обслуживания.** Для сокращения потерь времени, связанных с переключением программ, и обеспечения приоритета для коротких запросов (программ) используют дисциплину обслуживания с несколькими очередями (рис. 13.4). Приоритет очереди убывает с возрастанием ее номера  $m$  ( $0 \leq m \leq k$ ). Вновь поступающий в СРВ запрос ставится в конец очереди, имеющей старший приоритет ( $m=0$ ).

В такой системе следующим подлежит обслуживанию запрос из начала очереди с номером  $m$ , если очереди с меньшими номерами (с более высоким приоритетом) пустые. Если за выделенный программе, находившейся в очереди  $m$ , квант времени ее обработка не заканчивается, программа прерывается и переходит в конец очереди с номером  $m+1$ . Выделяемый программе квант времени возрастает с увеличением номера очереди обычно по правилу

$$\Delta_m = 2^m \Delta,$$

где  $\Delta_m$  — квант, выделяемый программе из  $m$ -й очереди ( $0 \leq m \leq k$ );  $\Delta$  — квант для программы из нулевой очереди. Обычно  $\Delta = 200 \div 500$  мс.

Программы из последней очереди обслуживаются до конца без прерывания. Если во время обслуживания программы из очереди  $m$  в очереди с большим приоритетом появляется новая программа, то после окончания текущего кванта  $\Delta$  обрабатываемая программа прерывается и возвращается в начало своей очереди, с тем чтобы впоследствии дополучить недоданное по отношению к значению  $2^m \Delta$  время.

Чтобы избежать недопустимо долгого ожидания для больших программ, приоритет делается зависящим от времени ожидания. Если ожидание превысит некоторое установленное значение (например, 60 с), программа перейдет в следующую очередь с меньшим номером. Таким образом, в рассматриваемой системе

приоритетные соотношения между запросами устанавливаются в самом процессе выполнения программ (*динамические приоритеты*).

Описанный механизм распределения времени обладает определенными адаптивными свойствами: программы с большей длительностью обработки вытесняются в очереди с большими номерами. Этот процесс можно сделать более эффективным, если при вводе запросов сразу распределять запрашиваемые программы по очередям, используя для этого некоторые оценки.

В первом приближении можно считать, что продолжительность выполнения программы пропорциональна ее длине (числу байт). По меньшей мере от длины программы прямо зависит время, затрачиваемое на передачу программ между оперативной и внешней памятью при переключении ее активности.

В алгоритме планирования Ф. Корбатто, реализованном в одной из первых СРВ, программа сразу поступает в очередь, имеющую номер

$$i = \lfloor \log_2 (l_p / l_{\Delta} + 1) \rfloor,$$

где  $l_p$  — число байт в программе;  $l_{\Delta}$  — число байт, которое может быть передано из оперативной памяти во внешнюю или обратно за время  $\Delta$ ; от выражений, стоящих в квадратных скобках, берется целая часть. В остальном дисциплина обслуживания соответствует описанной выше.

Процесс взаимодействия пользователя с СРВ показан на рис. 13.5. Пользователь со своего пульта вводит запрос (требо-

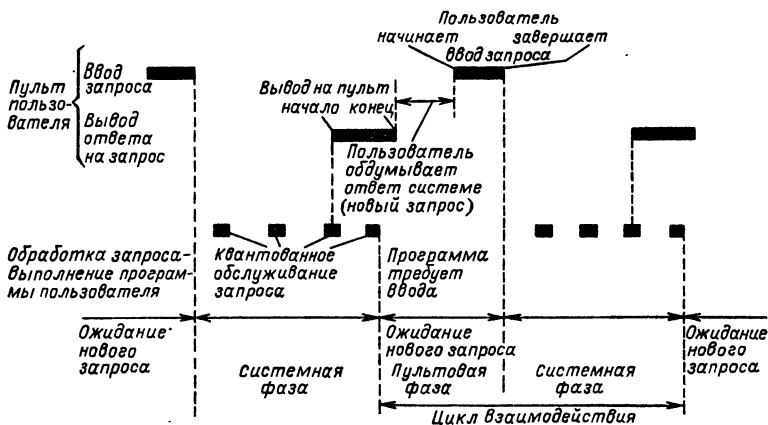


Рис. 13.5. Взаимодействие пользователя с СРВ

вание на выполнение некоторой программы и указание на используемые исходные данные).

Программа пользователя обслуживается в режиме разделения времени в соответствии с принятой в системе дисциплиной квантованного обслуживания до того момента, когда или обработка программы полностью завершится и произойдет вывод результата обработки на пульт пользователя, или программа дойдет до точки, когда требуются от пользователя новые указания или данные. В первом случае пользователь ставит системе новую задачу, во втором сообщает дополнительные данные, позволяющие системе продолжать обработку его программы.

### **13.4. Системные программные средства мультипрограммных ЭВМ общего назначения**

Эффективное функционирование ЭВМ общего назначения в различных режимах, их техническое обслуживание, работа пользователей по подготовке программ и их отладке, различные формы взаимодействия пользователей с ЭВМ обеспечиваются и поддерживаются комплексом системных программных средств (рис. 13.6).

Центральное место в этом комплексе занимают операционные системы (ОС). В состав ОС входят программные средства планирования вычислительного процесса (программы управления заданиями), организации вычислительного процесса, распределения ресурсов, управления вводом-выводом (программы супервизора), средства управления данными, а также средства подготовки и отладки программ (системные обрабатывающие программы).

Операционные системы позволяют осуществлять мультипрограммную работу в режимах пакетной обработки, разделения времени, реального времени, диалоговом и др.

Происходит процесс расширения и усложнения функций, возлагаемых на ОС (программное обеспечение телеобработки, машинной графики, режимов виртуальной памяти, виртуальных машин, работы с новыми языками программирования и т. п.). Появляются новые версии ОС. Ряд функций операционных систем передается для реализации аппаратурным, главным образом, микропрограммным средствам.

Машины ЕС-1046 в настоящее время в основном работают под управлением операционных систем ОС-6ЕС, обеспечивающих режим виртуальной памяти и возможность использования множества пакетов прикладных программ, а также более новой ОС-7ЕС, главной особенностью которой является наличие средств поддержки режима виртуальных машин (см. § 13.7).



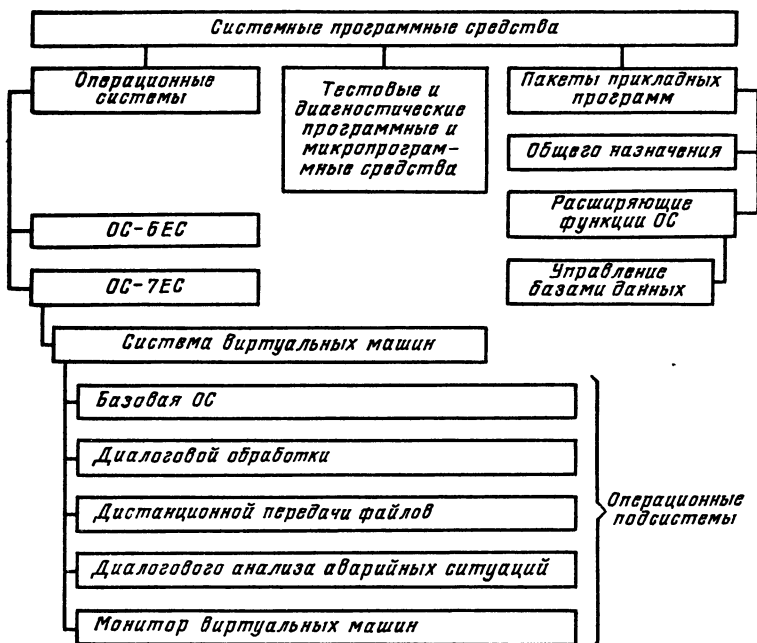


Рис. 13.6. Системные программные средства мультипрограммных ЭВМ общего назначения

Эти программные средства объединены в систему виртуальных машин, в которой, как это видно на рис. 13.6, можно выделить ряд операционных подсистем.

*Базовая система* ориентирована на решение задач в мультипрограммном пакетном режиме.

*Подсистема дистанционной передачи файлов* организует передачу файлов между удаленными терминалами и виртуальными машинами.

*Подсистема диалоговой обработки* представляет возможность пользователю в диалоговом режиме со своего терминала составлять и документировать программы на языках программирования, принятых в ЕС ЭВМ, редактировать их, отлаживать и запускать на выполнение, получать результаты на своем терминале.

*Подсистема диалогового анализа* обрабатывает информацию об аварийных ситуациях при работе в режиме виртуальных машин.

Все приведенные выше операционные подсистемы сами по себе являются диалоговыми (кроме базовой), но однопользовательскими. При функционировании этих подсистем под управлением монитора виртуальных машин реализуется одновременная работа (конечно, на основе разделения времени) этих подсистем на нескольких виртуальных машинах.

### 13.5. Особенности структуры ЭВМ общего назначения

Рассмотрим особенности структуры ЭВМ общего назначения на примере ЭВМ ЕС-1046 [68], при этом основное внимание сосредоточим на структурных решениях, связанных с обеспечением мультипрограммного режима работы и повышением производительности машины.

Упрощенная структура ЭВМ общего назначения представлена на рис. 13.7. Рисунок подчеркивает выделение ряда функций управления операциями ввода-вывода в отдельную функциональную систему, реализуемую в виде совокупности каналов ввода-вывода.

Значительная степень освобождения процессора от управления вводом-выводом является необходимым условием для мультипрограммной работы машины (необходимым, но недостаточным). Должна быть обеспечена сбалансированность пропускных способностей процессора, памяти и системы ввода-вывода, причем в условиях, когда в процессорах новых моделей ЭВМ за счет эффективных структурных решений (о них речь пойдет ниже) происходит значительный рост быстродействия, уменьшается продолжительность машинного такта (в ЭВМ ЕС-1046 она составляла 100 нс, а в более производительных машинах — еще меньше).

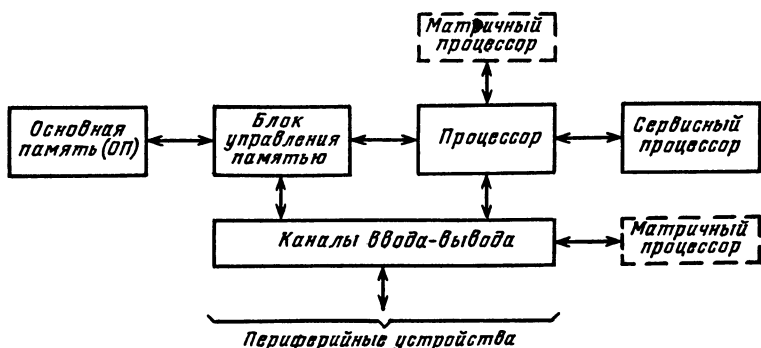


Рис. 13.7. Структура ЭВМ общего назначения (ЕС-1046)

Это приводит к необходимости (если иметь в виду систему ввода-вывода) повышать пропускную способность каналов и интерфейса ввода-вывода (см. гл. 11) и увеличивать общее число каналов. В ЭВМ ЕС-1046 шесть каналов, из них два байт-мультиплексных и четыре блок-мультиплексных с суммарной пропускной способностью до 10 Мбайт/с. В более быстродействующих моделях ЕС ЭВМ используются свыше десяти каналов ввода-вывода.

Важные и сложные функции, в том числе по обеспечению указанной выше сбалансированности, ложатся на блок управления памятью: организация в режиме многоабонентного обслуживания обмена между памятью, основными блоками процессора и каналами; защита областей памяти, выделенных одним программ, от несанкционированного вторжения в них других программ; организация функционирования сверхоперативной буферной памяти; реализация виртуальной памяти (см. гл. 14).

Сама ОП ЭВМ ЕС-1046 имеет следующие характеристики: емкость 8 или 16 Мбайт (построена на СБИС, содержащих 16 Кбит в корпусе); ширина выборки 16 байт, цикл обращения не превышает 0,7, а время выборки 0,55 мкс; автоматически корректируются одиночные ошибки (см. гл. 12).

Сложность аппаратуры, программных средств и режимов работы ЭВМ общего назначения потребовала включения в их состав средств, облегчающих и делающих более эффективными процедуры взаимодействия с машиной оператора или инженера по ее эксплуатации, оснащения ЭВМ аппаратурно-программными средствами поддержки эксплуатационного обслуживания (системы автоматического контроля, восстановления, диагностирования и др.). Важное место среди этих средств занимает специализированный сервисный процессор, который благодаря наличию в его составе дисплея, пультового ЗУ, оперативной памяти, печатающего устройства становится «интеллектуальным» пультом управления машины, позволяющим оператору (инженеру) выполнять разнообразные процедуры: включение и отключение напряжения питания, первоначальную загрузку микропрограмм в УП, установку режима работы, профилактические испытания, диагностирование неисправностей и др. (см. гл. 12, а также [27]).

Одним из важных путей повышения производительности ЭВМ общего назначения является включение в ее состав специализированных (проблемно-ориентированных) процессоров, предназначенных для решения задач определенного класса или выполнения определенных вычислительных процедур. Примером такого сопроцессора служит матричный процессор, ориентированный (благодаря широкому использованию конвейеризации

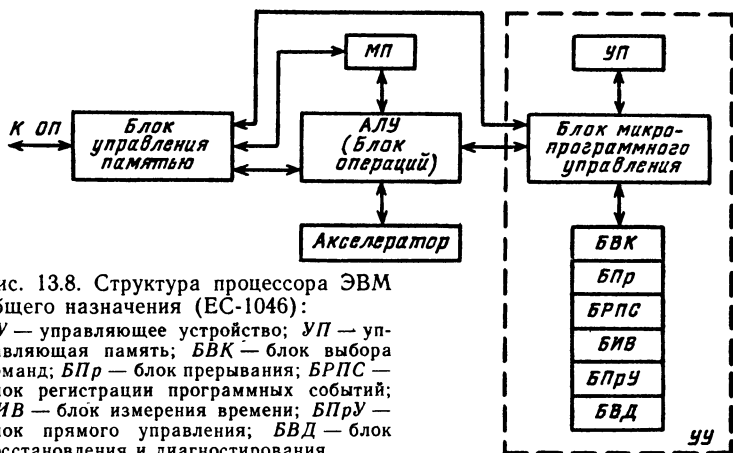


Рис. 13.8. Структура процессора ЭВМ общего назначения (ЕС-1046):

УУ — управляющее устройство; УП — управляющая память; БВК — блок выбора команд; БПр — блок прерывания; БРПС — блок регистрации программных событий; БИВ — блок измерения времени; БПрУ — блок прямого управления; БВД — блок восстановления и диагностирования

вычислений и введению специальных команд) на выполнение матричных и векторных операций. Специализированные сопроцессоры обычно подключаются, как это следует из рис. 13.7, либо к блок-мультиплексному каналу через интерфейс ввода-вывода, либо непосредственно к процессору в виде специализированного блока АЛУ.

Для более подробного рассмотрения особенностей структуры ЭВМ общего пользования следует обратиться к структуре ее процессора, представленной на рис. 13.8.

В машинах общего назначения процессор представляет собой объединяемую в один конструктивный модуль совокупность устройств (блоков), выполняющих разнообразные функции, обеспечивающие обработку данных, программное управление вычислительным процессом, взаимодействие устройств машины, обработку прерываний, обмен информацией с ОП.

Группируем эти устройства (блоки) процессора с учетом их основных функций, что можно сделать лишь достаточно условно из-за сложности их взаимодействия.

### Средства обработки данных

*Арифметическо-логическое устройство* выполняет арифметические операции с двоичными числами с фиксированной точкой, шестнадцатичными (двоично-кодированными) числами с плавающей точкой, десятичными (двоично-кодированными) числами переменной длины, логические операции над логическими кодами фиксированной и переменной длины; операции над алфа-

витно-цифровыми полями; операции формирования исполнительных адресов операндов.

Арифметическо-логическое устройство содержит четырехбайтный и однобайтный блоки операций. Четырехбайтный блок с 32-разрядным сумматором служит для операций над двоичными числами с фиксированной и плавающей точками, а также над логическими кодами. Однобайтный блок с 8-разрядным сумматором выполняет операции над десятичными числами и алфавитно-цифровыми полями переменной длины (см. гл. 7).

*Местная память* (МП) процессора. Ячейки МП используются в качестве 16 общих регистров (хранящих, в частности, операнды и результаты операций многих команд, значения базовых адресов и индексов), 4 регистров с плавающей точкой, регистров для промежуточных результатов (рабочая память), управляющих регистров для хранения данных блока восстановления и диагностики.

Часть МП служит в качестве местной памяти каналов ввода-вывода (хранит управляющие слова канала для устройств активных подканалов).

В ЭВМ ЕС 1065 МП имеет емкость 128 4-байтных слов, времена цикла обращения 50, а выборки 25 нс, т. е. более чем на порядок меньше, чем в ОП.

*Акселератор* служит для уменьшения времени выполнения 36 длинных команд (умножения, сдвига, двоичного деления, преобразования, упаковки и распаковки, некоторых пересылочных и логических операций).

При использовании акселератора выборка команд и подготовка операндов, а также запись результата производятся микропрограммой процессора, соответствующая микрокоманда которой запускает акселератор на выполнение предписанной командой операции.

Акселератор имеет собственную быстродействующую управляющую память с циклом 60 нс, которая хранит микропрограммы («пикопрограммы» или «нанопрограммы») акселератора. Таким образом, применительно к управлению операциями акселератора реализуется двухуровневое микропрограммирование (микропрограммы в УП процессора — пикопрограммы в УП акселератора).

Уменьшение времени выполнения операций достигается в акселераторе на основе применения конвейерных алгоритмов и использования постоянных программируемых памяти (ППЗУ) для табличных преобразований.

Акселератор уменьшает время выполнения отдельных операций в 2—4 раза, при этом общая производительность процессора увеличивается на 15—20 %.

Для многих машин общего назначения характерным является использование смешанного схемного (на основе «схемной» логики) и микропрограммного (с хранимой в памяти логикой) управления. Так, в процессоре ЭВМ ЕС-1046 реализовано микропрограммное, а, например, в каналах ввода-вывода — комбинированное схемно-микропрограммное управление, причем в каналах под микропрограммным управлением (соответствующие микропрограммы хранятся в УП процессора) выполняются те процедуры, в которых каналы используют аппаратуру процессора (передача информации между каналами и ОП с использованием цепей обмена данными процессора с ОП; операции над управляющей информацией канала с привлечением средств АЛУ). Выполнением процедур работы каналов, не связанных с использованием аппаратуры процессора, управляют собственные схемные управляющие средства каналов.

*Блок микропрограммного управления* (БМУ) осуществляет в режиме разделения времени управление работой процессора и некоторыми процедурами каналов ввода-вывода, используя для этого микропрограммы, находящиеся в управляющей памяти, выдает предписанные микрокомандой управляющие сигналы в соответствующие блоки процессора, формирует адрес следующей микрокоманды, обрабатывает прерывание микропрограмм. Запросы каналов на обслуживание со стороны БМУ имеют больший приоритет по сравнению с процессором, а среди каналов устанавливаются свои приоритетные соотношения. При прерывании микропрограммы в БМУ запоминается адрес следующей подлежащей выполнению микрокоманды.

В ЭВМ ЕС-1046 микрокоманда имеет длину 72 разряда, используется горизонтально-вертикальное микропрограммирование. В формате микрокоманды есть поле, задающее адрес следующей микрокоманды. Два младших разряда этого адреса в общем случае устанавливаются в зависимости от значений оповещающих сигналов, вырабатываемых по результатам проверки определенных условий, задаваемых специальными полями микрокоманд.

*Управляющая память* (УП) хранит микропрограммы, ее емкость составляет 8192 72-разрядных микрокоманд. Управляющая память является полностью перезагружаемой (время считывания 55, а записи 75 нс). Загрузка микропрограмм в УП (в том числе начальная) производится с пультного ЗУ. Возможна также загрузка УП микропрограммами из ОП, что создает основу для динамического микропрограммирования (см. §13.8).

*Блок выборки команд (БВК)* производит с помощью буферных регистров предварительную (опережающую) выборку трехкомандных слов, их распаковку и хранение команд, формирует продвинутый адрес команды.

*Блок (система) прерываний* (см. § 9.18).

*Блок регистрации программных событий (БРПС)* облегчает отладку программ, оповещая при помощи прерывания о некоторых событиях, возникающих при выполнении программы. Возможна регистрация следующих программных событий: успешное выполнение команды перехода, выборка команды из заданной области ОП, изменение заданной области ОП, изменение содержимого заданных общих регистров. Программа, заставляя соответствующие коды в управляющие регистры и единицу в разряд маски регистрации программных событий в ССП, задает подлежащие регистрации события и контролируемые при этом области памяти и общие регистры.

*Блок измерения времени (БИВ)* формирует коды даты и суточного времени, может по заданию программы измерять промежутки времени и вырабатывать внешнее прерывание по истечении заданного интервала времени.

*Блок прямого управления (БПрУ)* позволяет производить прямой обмен данными между процессорами в многомашинной системе или между процессором и внешним по отношению к ЭВМ оборудованием (см. гл. 15).

*Блок восстановления и диагностики* — см. гл. 12.

### *Средства блока управления памятью*

К средствам блока управления памятью относятся память ключей защиты, сверхоперативная буферная память (кэш-память) и блок динамического преобразования адресов при реализации виртуальной памяти, построение и функционирование которых рассмотрено в гл. 14.

В заключение отметим, что ЭВМ общего назначения ЕС-1046 обладает производительностью 1300 тыс. команд/с при решении научно-технических задач и 650 тыс. команд/с при решении задач обработки данных.

## **13.6. Принцип виртуализации ресурса. Система виртуальных машин**

В современной вычислительной технике широко используется *принцип виртуализации* (виртуальный — означает кажущийся) ресурса вычислительной установки в целях упрощения (в том числе унификации процедур) взаимодействия с ней пользовате-

ля. Виртуализация ресурса освобождает пользователя от необходимости учета ограничений, связанных с характеристиками реального ресурса (например, размера емкости реальной памяти), а также ограничений, вызываемых многопользовательским режимом работы (режимом коллективного пользования). Пользователь получает возможность игнорировать особенности аппаратуры и функционирования реального ресурса (прозрачность виртуального ресурса).

Виртуализация может заключаться в приписывании ресурсу (памяти, периферийному устройству) таких характеристик и свойств, которыми он в действительности не располагает, но которыми тем не менее пользователю разрешается пользоваться. При этом на аппаратурно-программные средства ложится отображение виртуальных характеристик и свойств на реальные, присущие конкретной ВС. Примерами такой виртуализации ресурса являются «виртуальная память» (см. § 14.5) и «виртуальный терминал» в вычислительных сетях (см. гл. 16). Заметим, что реализуемая в ЕС ЭВМ унификация командных средств и процедур взаимодействия ядра ЭВМ с периферийными устройствами фактически достигается на основе виртуализации периферийных устройств.

Виртуализация может также состоять в выделении программе или пользователю части рабочего времени процессора или периферийного устройства или части емкости носителя информации ЗУ на дисках или лентах, если эта часть ресурса воспринимается и с ней манипулируют, как со своим процессором, своим устройством ввода-вывода, своим дисковым ЗУ и т. д.

Примером таким образом формируемых периферийных виртуальных устройств является организация с помощью дисковых или ленточных ЗУ системного ввода и вывода с накоплением на них наборов данных, поступающих с перфокартного или иного устройства ввода («виртуальное устройство ввода»), а также данных, подлежащих выдаче на печатающее устройство («виртуальное печатающее устройство»).

### *Система виртуальных машин*

Наиболее развитой формой реализации принципа виртуализации ресурса является организация системы виртуальных машин (СВМ) в одной реальной машине [68], чем достигается повышение уровня мультипрограммирования и эффективности использования всех ее ресурсов несколькими пользователями, облегчается их взаимодействие с машиной.

При обычном мультипрограммном режиме не все ресурсы машины доступны пользователю (его программе). Так, он не



может использовать привилегированные команды, производить пультовые операции и т. д.

Для системы виртуальных машин в составе реальной ЭВМ при помощи специальных аппаратурно-программных средств формируются несколько параллельно во времени функционирующих «*виртуальных машин*» (ВМ), и пользователь, получив в свое распоряжение одну из них, имеет возможность сам задавать основные пультовые операции, загружать в свою ВМ операционную систему по своему выбору и программу своей задачи, использовать привилегированные команды.

Режим виртуальных машин в ЕС ЭВМ реализуется под управлением программного пакета «Система виртуальных машин» (см. рис. 13.6), содержащего монитор виртуальных машин, который на технических средствах реальной ЭВМ моделирует процессор, основную память, пульт оператора, каналы ввода-вывода, периферийные устройства виртуальной машины, организует моделирование привилегированных команд, присутствующих в программах виртуальных машин<sup>1</sup>.

Создание виртуальной машины фиксируется оператором ЭВМ по соответствующему запросу с местного или удаленного терминала, с которого затем пользователь может загружать в нее свои программы и запрашивать выполнение пультовых операций.

Организация режима ВМ поясняется на рис. 13.9 [68], который подчеркивает возможность одновременной работы виртуальных машин с разными операционными системами и другими операционными средствами. К последним, в частности, относятся подсистемы дистанционной передачи файлов, диалоговой обработки (занесение в ВМ программ, написанных на языках программирования, их трансляция и исполнение), диалогового анализа и учета аварийных ситуаций при функционировании ВМ.

Реализация нескольких виртуальных машин на одной реальной ЭВМ сопряжена с потерями времени, связанными с эмуляцией монитором программных прерываний, порождаемых привилегированными командами и командой «Вызов супервизора» в программах виртуальных машин, приводящих к переходу реальной машины в режим супервизора. Эти потери времени удастся существенно сократить путем поддержки дополнительными микропрограммными средствами некоторых функций системы виртуальных машин (см. § 13.7).

---

<sup>1</sup> Привилегированная команда вызывает прерывание виртуальной машины, которая продолжит работу после завершения моделирования запрашиваемой этой командой процедуры.

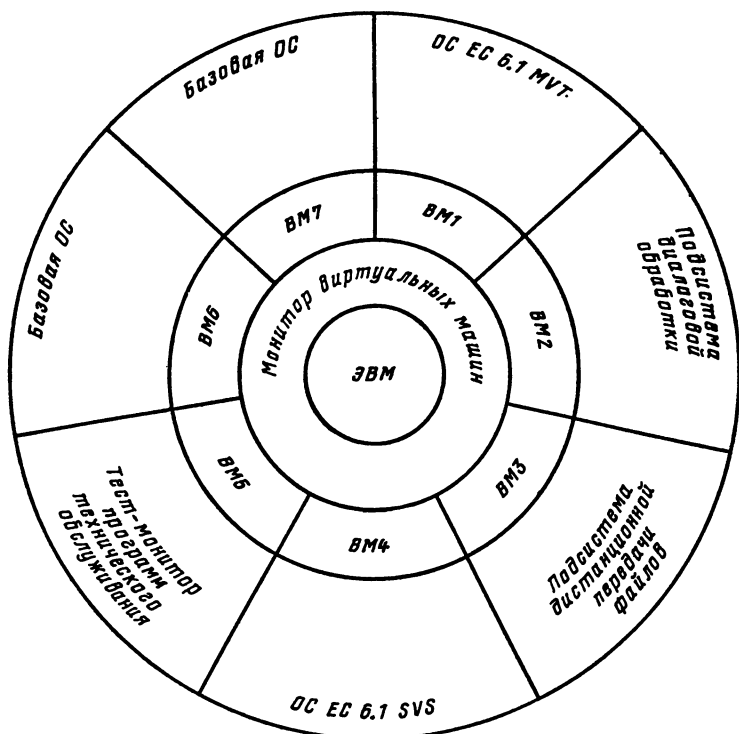


Рис. 13.9. Организация режима виртуальных машин

### 13.7. Динамическое микропрограммирование — средство динамической модификации архитектуры ЭВМ

Микропрограммирование называют динамическим, если имеется возможность в ходе вычислительного процесса производить программно-управляемую загрузку микропрограмм из ОП в управляющую память (УП) процессора. Для этого необходимо, чтобы УП была полностью или частично перезагружаемой, т. е. допускающей изменение содержащейся в ней информации.

Рассмотрим использование динамического программирования на примере машины ЕС-1046 [68]. В этой ЭВМ вся УП (емкостью 8 Кбайт) является перезагружаемой, при этом 3/4 ее емкости используется для хранения загружаемых при «начальной загрузке» с пультавого ЗУ (ленточного или с гибким диском) микропрограмм, реализующих стандартные свойства архитектуры ЕС ЭВМ, а 1/4 емкости резервирована для динамически загружаемых из ОП микропрограмм специальных про-

цедур, варьируемых в зависимости от режима работы машины и характера решаемых задач. Объем этих микропрограмм достаточно велик, и все они не могут разместиться в «переменной» части УП. Поэтому они хранятся на магнитной ленте и в нужных случаях через ОП загружаются в УП.

Микропрограммы режима динамического микропрограммирования в основном используются для аппаратурной (микропрограммной) реализации:

некоторых функций системы виртуальных машин, в том числе монитора виртуальных машин, что снижает издержки на организацию функционирования виртуальной машины;

часто встречающихся в прикладных задачах вычислительных процедур, чем достигается значительное сокращение времени их выполнения, а следовательно, и ориентация машины на определенный класс прикладных задач.

Поскольку путем динамического микропрограммирования аппаратурные средства ЭВМ приобретают новые свойства, можно говорить о динамическом (в ходе вычислительного процесса) изменении архитектуры машины.

Микропрограммная реализация некоторых функций системы виртуальных машин имеет целью освободить монитор виртуальных машин от необходимости при моделировании этих функций переводить с помощью прерывания программы реальную машину в режим супервизора. С помощью микропрограммных средств на виртуальной машине (без прерывания реальной ЭВМ и перевода ее в режим супервизора или со значительным сокращением времени ее пребывания в этом режиме) выполняются команды «Вызов супервизора» и привилегированные команды («Установить маску системы», «Загрузка ССП», «Установить ключ программы» и др.). При этом потери времени на эти операции уменьшаются в среднем примерно в 50 раз.

Микропрограммными средствами реализуются следующие группы вычислительных операций: вычисление элементарных функций (синуса и косинуса); векторные операции (сложение векторов, скалярное произведение векторов, поэлементное умножение векторов и др.); определение количества нулей в двоич-

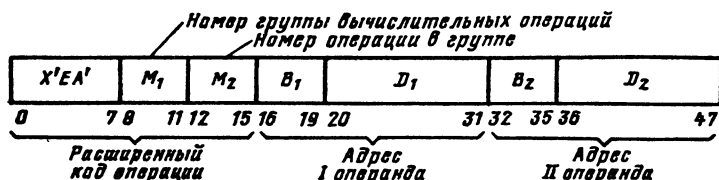


Рис. 13.10. Формат специальной команды ASOP

ном коде; поиск минимального байта; операции матричного процессора в его отсутствие в составе ЭВМ (прямое и обратное быстрые преобразования Фурье, разностные уравнения и др.).

Эти операции задаются с помощью специально для этой цели введенной команды ASOP с кодом операции X'EA', имеющей формат, представленный на рис. 13.10.

Микропрограммная реализация группы операций, соответствующих командам матричного процессора, позволяет для многих из них достигнуть почти половины скорости выполнения операций в матричном процессоре. Для других групп вычислительных операций время их выполнения при микропрограммной поддержке уменьшается примерно в 3 раза.

В большинстве случаев в переменную область УП загружаются микропрограммные средства поддержки виртуальных машин вместе со средствами поддержки нужной группы вычислительных операций.

### **Контрольные вопросы**

1. Какое содержание вкладывается в термин «ЭВМ общего назначения»? Каковы особенности архитектуры этих машин, поддерживающие их «общее назначение»?

2. Какова цель организации мультипрограммного режима работы ЭВМ и какими аппаратными и программными средствами обеспечивается реализация этого режима?

3. Как в системах разделения времени обеспечивается приоритетная обработка коротких запросов?

4. Сравните системы разделения времени и системы виртуальных машин по назначению и способу функционирования.

5. Что такое динамическое микропрограммирование и как на его основе реализуется динамическая модификация архитектуры ЭВМ? Приведите примеры.

## ПРИНЦИПЫ ОРГАНИЗАЦИИ МНОГОУРОВНЕВОЙ СИСТЕМЫ ПАМЯТИ В МУЛЬТИПРОГРАММНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

### 14.1. Проблемы организации памяти мультипрограммных систем

Память современных вычислительных систем имеет иерархическую многоуровневую структуру. Чем выше уровень, тем выше быстродействие соответствующей памяти, но меньше ее емкость. К верхнему уровню относятся запоминающие устройства, с которыми процессор непосредственно взаимодействует в процессе выполнения программы — основная или оперативная память (ОП) (см. гл. 4).

Сравнительно небольшая емкость ОП (64 — 16 000 Кбайт) компенсируется практически неограниченной емкостью внешних запоминающих устройств на магнитных дисках и лентах (сотни миллионов или миллиарды байт). Однако эти устройства сравнительно медленные, и время обращения за данными для дисков составляет десятки миллисекунд, а для лент может достигать сотен секунд. Поэтому вычислительный процесс должен протекать с возможно меньшим числом обращений к внешним запоминающим устройствам и максимально возможным использованием ОП.

Быстродействие ОП часто оказывается недостаточным для обеспечения требований, предъявляемых к скорости работы ЭВМ. Это проявляется в несоответствии пропускных способностей процессора и ОП. Возникающая проблема выравнивания их пропускных способностей решается путем использования *буферных памяти* небольшой емкости и повышенного быстродействия, хранящих команды и данные, относящиеся к обрабатываемому участку программы.

Оперативная память является наиболее дефицитным ресурсом в вычислительных системах, которым надо пользоваться экономно и эффективно. Проблема усложняется при переходе к мультипрограммным системам, так как в них ОП одновременно используют несколько программ (заданий). В таких системах необходимо исключить несанкционированное воздействие одних программ на другие. Это достигается с помощью механизма *защиты памяти*.

Необходимо обеспечить доступ к ОП со стороны нескольких ее абонентов, в качестве которых выступают основные блоки процессора, каналы ввода-вывода и др.

Эффективное распределение ресурса памяти между программами не может быть статическим, т. е. не может производиться предварительно до пуска программы. В процессе обработки программ потребности в ресурсе памяти отдельных программ изменяются, что заранее не может быть учтено. Необходимо распределять память между программами динамически непосредственно в ходе вычислительного процесса, т. е. осуществлять *динамическое распределение памяти*. При этом должна обеспечиваться возможность независимой работы программистов над своими программами, подлежащими мультипрограммной обработке. Динамическое распределение памяти не должно приводить к дроблению ее свободного пространства — *фрагментации памяти*, затрудняющему ее использование. Это достигается организацией *одноуровневой виртуальной памяти*, допускающей адресацию на все адресное пространство. Размер его определяется количеством разрядов, которые могут быть использованы для представления адреса.

#### **14.2. Согласование пропускных способностей процессора и памяти ЭВМ. Кэш-память**

Непрерывный рост производительности (скорости работы) ЭВМ, вызываемый потребностями их применений, проявляется, в первую очередь, в повышении скорости работы процессов, достигаемой использованием новых, более быстродействующих электронных схем, а также специальных архитектурных решений — конвейерная и векторная обработка данных и др. Быстродействие оперативной памяти также растет, но все время отстает от быстродействия аппаратурных средств процессора, в значительной степени потому, что одновременно происходит опережающий рост ее емкости, что делает более трудным уменьшение времени цикла работы памяти.

Без согласования пропускных способностей процессора и памяти невозможно в машине реализовать производительность, соответствующую быстродействию процессора.

Преодолеть указанное противоречие и согласовать пропускные способности памяти и процессора помогают специальные структурные решения [36, 49].

*Конвейеризация процедур цикла выполнения команды* (рабочего цикла машины), в простейшем случае — выполнение параллельно во времени операции в АЛУ с выборкой из памяти следующей команды (см. § 9.20).

«Расслоение» ОП путем многомодульного построения с «верной» («чередующейся») адресацией, при которой смежные адреса информационных единиц, соответствующих ширине выборки (слово, двойное слово и т. п.), принадлежат разным модулям. При этом повышается пропускная способность ОП за счет перекрытия во времени обращений к разным модулям памяти.

При отсутствии расслоения время выборки из ОП  $m$  чисел составляет [49]

$$t_{\text{выб}} = t_{\text{счит}} + (m - 1) t_{\text{ц}},$$

а при наличии расслоения и при задержке на один такт обращения к очередному модулю ОП

$$t_{\text{выб}} = t_{\text{счит}} + (m - 1) t_{\text{т}},$$

где  $t_{\text{выб}}$  — время выборки;  $t_{\text{счит}}$  — время считывания слова в ОП;  $t_{\text{ц}}$  — продолжительность цикла обращения к ОП;  $t_{\text{т}}$  — продолжительность машинного такта. В силу того что  $t_{\text{т}} \ll t_{\text{ц}}$ , время выборки при расслоении ОП существенно уменьшается.

*Буферизация* — использование включенных между процессором и ОП существенно более чем ОП быстродействующих буферных памяти сравнительно небольшой емкости.

На рис. 14.1 показана структура процессора, содержащая буферную память команд и буферную память операндов.

Представленные на рис. 14.1 буферные памяти скрыты от программиста в том смысле, что он не может их адресовать и может даже не знать об их существовании. Поэтому они получили название кэш-памятей<sup>1</sup>. Структура некоторых ЭВМ содержит объединенную кэш-память для фрагментов программ и групп данных, при этом в ряде случаев наряду с кэш-памятью (в дальнейшем для краткости именуемой кэш) сохраняется небольшой буфер на несколько команд.

При обращении процессора к ОП для считывания в кэш передается блок информации, содержащий нужное слово. При этом происходит опережающая выборка, так как высока вероятность того, что ближайшие обращения будут происходить к словам этого же блока, уже находящимся в кэш. Это приводит к значительному уменьшению среднего времени, затрачиваемого на выборку данных.

Эффективность кэш, зависящая от ее емкости, размера блока, соотношения времен считывания слова из кэш и блока из ОП проявляется в уменьшении среднего времени, затрачиваемо-

<sup>1</sup> Кэш — от английского cache — тайник.

го на выборку слова данных и определяемого выражением

$$t_{\text{счит.ср}} = t_{\text{кэш}} + t_{\text{опбл}}/k_{\text{ср}},$$

где  $t_{\text{кэш}}$  — время считывания слова из кэш;  $t_{\text{опбл}}$  — время считывания блока из ОП;  $k_{\text{ср}}$  — среднее число обращений к кэш между двумя последовательными обращениями к ОП.

Следует отметить, что рас-  
слоение памяти существенно  
уменьшает  $t_{\text{опбл}}$ , позволяя при  
этом с задержками на один  
такт считывать группу слов из  
ячеек ОП с последовательными адресами.

Можно выделить два типа кэш-памяти:

а) с запоминанием новой информации одновременно в кэш и в ОП («сквозное запоминание»), при этом всегда в ОП есть последняя копия хранящейся в кэш информации. Однако в этом случае длинный цикл ОП снижает производительность процессора;

б) с запоминанием новой информации только в кэш и копированием ее в ОП только при передаче в другие устройства или при вытеснении из кэш<sup>1</sup>.

Рассмотрим организацию кэш-памяти, ориентируясь на принятые в ЕС ЭВМ проектные решения (рис. 14.2) [48, 68].

Кэш-память представляет собой достаточно сложное устройство, что связано с тем, что она должна содержать средства, определяющие, находится ли в кэш блок со словом, которое запрашивает процессор. Эта задача решается применением быстродействующей матрицы адресов (МА).

КЭШ в ЭВМ ЕС-1046 имеет емкость 16 Кбайт (ширина выборки 72 разряда) и время доступа 35 нс, соизмеримое с продолжительностью такта процессора. Обмен с ОП производится 16-байтными блоками (содержат два двойных слова). В матрицу адресов, имеющую емкость 256 60-разрядных слов и время выборки около 17 нс, заносятся сведения об адресах (в ОП) блоков, помещаемых в кэш.

Оперативная память условно разбита на горизонтальные ряды и вертикальные колонки. Ряд содержит 256 16-байтных блоков (всего 4 Кбайт). При емкости ОП 16 Мбайт число рядов составляет 4096. Сама кэш состоит из четырех отделений (I—

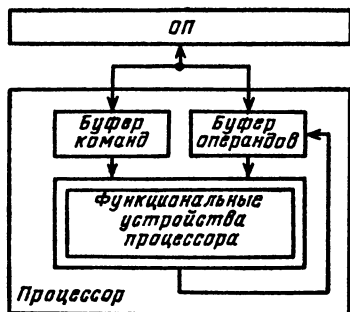
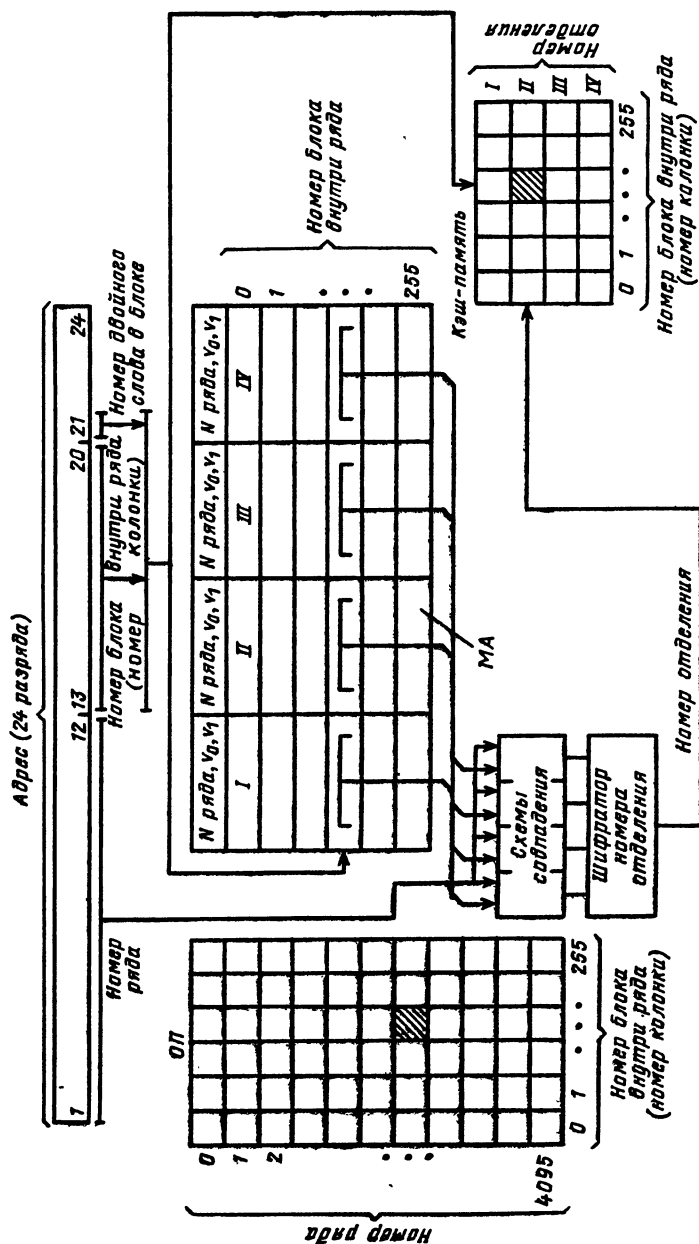


Рис. 14.1. Буферные памяти команд и операндов

<sup>1</sup> В англоязычной литературе эти типы кэш получили название store-through и store-in-cashe.





IV), в каждом из которых могут быть размещены 256 блоков данных. Каждому отделению кэш поставлено в соответствие определенное отделение МА. Таким образом, кэш и МА содержат одинаковое число ячеек. Только в ячейках кэш хранятся сами информационные блоки, а в соответствующих ячейках МА — сведения об адресах, точнее, номерах рядов блоков в ОП, а также разряды действительности  $v_0$ ,  $v_1$ . Кроме того, имеется еще вспомогательная матрица (на рис. 14.2 не показана), по существу, являющаяся частью МА, в которой формируются два контрольных разряда исправности отделений I—II и III—IV кэш и МА, а также разряды активности, выделяющие наиболее пассивное отделение.

Число вертикальных колонок в ОП и кэш одно и то же. Всем блокам, имеющим один и тот же номер внутри ряда (разряды 13—20 адреса в ОП), соответствует определенная колонка в кэш и определенная строка в МА.

В кэш, представленной на рис. 14.2, в колонке могут помещаться не более четырех блоков, принадлежащих разным рядам, но имеющих один и тот же номер внутри ряда. При таком выполнении устройства достаточно в ячейке МА фиксировать номер ряда (разряды 1—12 адреса), к которому принадлежит помещаемый в кэш блок информации. Ячейка МА содержит также разряды действительности  $v_0$  и  $v_1$ , устанавливаемые в 1 в зависимости от того, к какому из двух двойных слов происходит обращение (определяется разрядом 21 адреса).

При любом обращении процессора в ОП для записи или считывания производится проверка наличия соответствующего блока в кэш. По содержащемуся в адресе номеру блока из МА считывается строка с четырьмя номерами рядов, которые в схемах сравнения сопоставляются с номером ряда в адресе обращения. При обнаружении совпадения (блок находится в кэш) шифратор формирует номер отделения кэш, в котором в позиции, задаваемой в адресе номером блока (колонки), находится запрашиваемый блок. Производится обращение в кэш к адресуемому двойному слову из этого блока по считыванию или записи с установкой разряда действительности  $v_0$  и  $v_1$ . При этом дублирование записи новой информации в ОП производится или не производится в зависимости от того, к какому из указанных выше типов относится данная кэш.

Если нет совпадения номеров рядов, т. е. в кэш нет запрашиваемого блока, он считывается из ОП; адресуемое двойное слово поступает в процессор, а блок записывается в кэш в соответствующую номеру блока внутри ряда колонку, замещая в ней блок, к которому дольше всего не было обращений. Этот блок определяется по состоянию разрядов активности.

В ЭВМ ЕС-1046 применение кэш с временем выборки 2 машинных такта вместо 11 в ОП позволило повысить производительность машины почти в 2 раза.

### 14.3. Защита памяти

Если в памяти одновременно могут находиться несколько независимых программ, необходимы специальные меры по предотвращению или ограничению обращений одной программы к областям памяти, используемым другими программами. Программы могут содержать такие ошибки, которые, если этому не воспрепятствовать, приводят к искажению информации, принадлежащей другим программам. Последствия таких ошибок особенно опасны, если разрушению подвергнутся программы операционной системы. Другими словами, надо исключить воздействие программы пользователя на работу программ других пользователей и программ операционной системы.

Чтобы воспрепятствовать разрушению одних программ другими, достаточно защитить область памяти данной программы от попыток записи в нее со стороны других программ, а в некоторых случаях и своей программы (*защита от записи*), при этом допускается обращение других программ к этой области памяти для считывания данных.

В других случаях, например при ограничениях на доступ к информации, хранящейся в системе, необходимо иметь возможность запрещать другим программам производить как запись, так и считывание в данной области памяти. Такая *защита от записи и считывания* помогает отладке программы, при этом осуществляется контроль каждого случая выхода за область памяти своей программы.

Для облегчения отладки программ желательно выявлять и такие характерные ошибки в программах, как попытки использования данных вместо команд или команд вместо данных в собственной программе, хотя эти ошибки могут и не разрушать информацию.

Отметим следующие варианты дифференцированной защиты при различных операциях с памятью:

1) задается отношение к области памяти чужой программы, определяющее, относится защита памяти только к операции записи или к любому обращению в память;

2) задается одно из следующих отношений к области памяти собственной программы;

а) разрешается доступ к данному блоку как для записи, так и для считывания;

б) разрешается только считывание;

в) разрешается обращение любого вида, но по адресу, взятому только из счетчика команд;

г) разрешается обращение по адресу из любого регистра, кроме счетчика команд.

Если нарушается защита памяти, исполнение программы приостанавливается и вырабатывается запрос прерывания по нарушению защиты памяти.

Защита от вторжения программ в чужие области памяти может быть организована различным образом, при этом реализация защиты не должна заметно снижать производительность машины и требовать слишком больших аппаратных затрат.

**Защита отдельных ячеек памяти.** В управляющих вычислительных комплексах, предназначенных для работы в АСУ ТП, необходимо обеспечить возможность отладки новых программ параллельно с функционированием находящихся в памяти рабочих программ, управляющих технологическим процессом. Это может быть достигнуто выделением в каждой ячейке памяти специального «разряда защиты». Установка 1 в этот разряд запрещает производить запись в данную ячейку.

В системах с мультипрограммной обработкой большого числа программ защищаются не отдельные ячейки, а области памяти или блоки, на которые делится память, при этом часто предусматривается возможность указывать для разных программ различные допустимые режимы обращения к отдельным областям или блокам памяти.

**Метод граничных регистров** (рис. 14.3) состоит во введении двух граничных регистров, указывающих верхнюю и нижнюю границы области памяти, куда программа имеет право доступа.

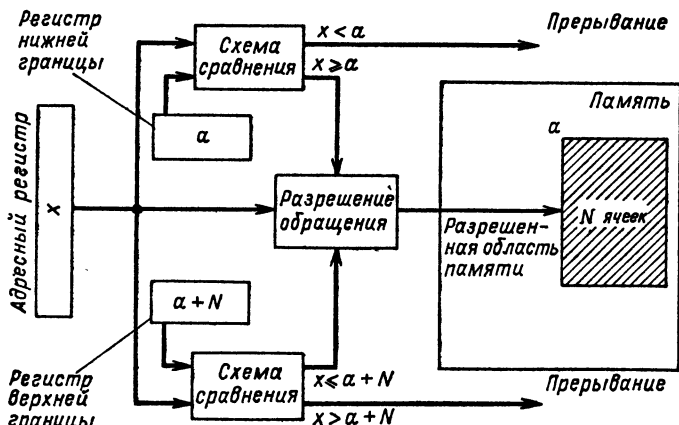


Рис. 14.3. Защита памяти с помощью граничных регистров

При каждом обращении к памяти проверяется, находится ли используемый адрес в установленных границах; при выходе за границы обращение к памяти подавляется и формируется запрос прерывания, передающий управление операционной системе. Содержание граничных регистров устанавливается операционной системой перед тем, как для очередной целевой программы начнется активный цикл. Если для динамичного распределения памяти используется базовый регистр, то он одновременно определяет и нижнюю границу. Верхняя граница подсчитывается операционной системой в соответствии с длиной программы в оперативной памяти.

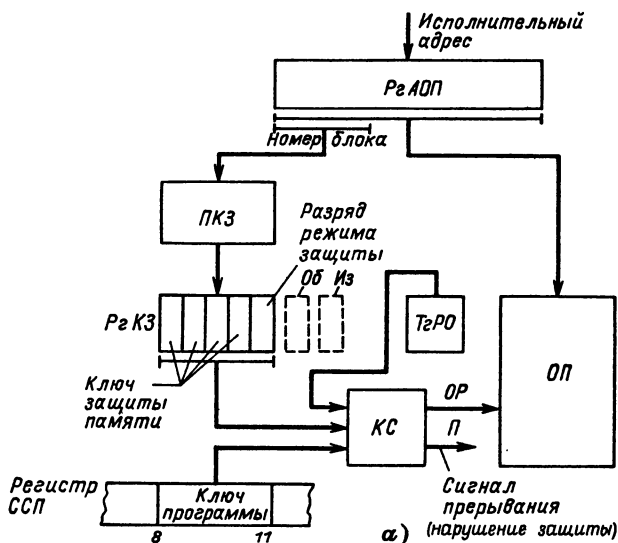
*Метод ключей защиты.* По сравнению с предыдущим данный метод является более гибким: он позволяет организовать доступ программы к областям памяти, расположенным не подряд.

Память в логическом отношении делится на одинаковые блоки. Каждому блоку памяти ставится в соответствие код, называемый *ключом защиты памяти*, а каждой программе, принимающей участие в мультипрограммной обработке, присваивается код *ключа программы*. Доступ программы к данному блоку памяти для чтения и записи разрешен, если ключи совпадают или один из них имеет код 0.

Метод ключей защиты используется в ЕС ЭВМ. Блоки содержат по 2048 байт. В мультипрограммном режиме могут одновременно обрабатываться до 16 (включая операционную систему) программ. Поэтому для исключения влияния программ друг на друга следует иметь возможность задавать 16 различных вариантов кодов ключей защиты памяти и ключей программы, для чего необходимы 4-разрядные коды. Четырехразрядный ключ программы указывается в специальном поле слова состояния программы (ССП), а при операциях ввода-вывода — в слове состояния канала (ССК). В ключе защиты памяти предусматривается дополнительный, пятый (младший), *разряд режима защиты*. Защита действует только при попытке записи в блок, если в этом разряде стоит 0, и при записи и считывании, если стоит 1. Коды ключей защиты памяти и ключей программы устанавливаются операционной системой с помощью специальных команд, относящихся к привилегированным операциям.

Коды ключей защиты памяти хранятся в специальной *памяти ключей защиты*, более быстродействующей, чем ОП.

Функционирование защиты памяти поясняется схемой на рис. 14.4. При обращении к памяти исполнительный адрес поступает в регистр адреса оперативной памяти *РзАОП*. Запускается цикл ОП, и одновременно группа старших разрядов кода исполнительного адреса, соответствующая номеру блока, к которому производится обращение, используется как адрес для вы-



Ключ защиты блока А	Разряд режима защиты	Ключ программы в ССП		
1101	0	1101	Запись в А, считывание из А	Разрешено (ОР=1)
1101	1	1101	Запись в А, считывание из А	
0011	0	1101	Запись в А, считывание из А	Запрещено (П=1), разрешено (ОР=1)
0011	1	1101	Запись в А, считывание из А	
0011	0	0000	Запись в А, считывание из А	Разрешено (ОР=1)
0011	1	0000	Запись в А, считывание из А	

б)

Рис. 14.4. Защита памяти при помощи ключей защиты. Структура защиты памяти (а) и различные ситуации в работе защиты (б)

борки из памяти ключей защиты  $ПКЗ$  и передачи на регистр ключа защиты памяти  $РзКЗ$  кода ключа защиты памяти, присвоенного операционной системой данному блоку. Комбинационная схема  $КС$  сравнивает ключ защиты памяти блока и ключ программы в регистре  $ССП$  или  $ССК$  и вырабатывает с учетом режима обращения (запись или считывание), указываемого триггером режима обращения  $ТзРО$ , и режима защиты (устанавливается разрядом режима в ключе защиты памяти) сигнал

«Обращение разрешено» ( $OP=1$ ) либо сигнал «Прерывание» ( $P=1$ ) по нарушению защиты.

Если ключи защиты блока памяти и программы совпадают или при любом ключе защиты памяти ключ программы имеет код 0000 (такой ключ программы присвоен программе операционной системы), то разрешены обращения к данному блоку для записи и считывания независимо от назначения разряда режима защиты. Обращения также разрешены при любом ключе программы, если ключ памяти имеет код 0000 (такой ключ обычно имеют блоки, хранящие стандартные подпрограммы).

#### **14.4. Организация работы памяти в режиме многоабонентного обслуживания**

В современных мультипрограммных ВС для повышения производительности реализуется независимая параллельная работа во времени процессора, байт-мультиплексного и селекторных (блок-мультиплексных) каналов и некоторых других устройств (например, таймера), которые асинхронно и независимо друг от друга формируют запросы на обращение к ОП. Более того, в процессоре имеется ряд параллельно работающих блоков, независимо друг от друга обращающихся к ОП. Например, такими блоками процессора могут являться блок контроля и диагностирования (БКД), блок адреса результата (БАР), блок центрального управления (БЦУ) и блок выборки команд (БВК).

Перечисленные выше устройства и блоки можно рассматривать как абоненты памяти. Возникает задача обеспечения работы ОП в режиме многоабонентного обслуживания. Этот режим должен осуществляться таким образом, чтобы эффективно использовалась для повышения общей пропускной способности памяти ее модульная организация, допускающая независимые обращения к отдельным модулям, обеспечивалась защита памяти, контроль считываемых и записываемых данных, контроль нарушения адресации.

Указанные выше функции реализуются блоком обращения к оперативной памяти (БОП). Рассмотрим принцип построения БОП [18].

Приоритеты запросов отдельных устройств и блоков на обслуживание со стороны ОП упорядочиваются согласно рис. 14.5. Каналы имеют более высокий приоритет, чем процессор. Это связано с опасностью потери информации при работе с движущимся носителем.

Как видно из рис. 14.5, связи БОП с абонентами, модулями ОП и памятью ключей защиты (ПКЗ) осуществляются коллективными и индивидуальными (радиальными) шинами.

Рис. 14.5. Многоабонентный режим работы ОП:

ПКЗ — память ключей защиты; ША и ШИ — соответственно адресные и информационные коллективные шины; И, А, У, К — индивидуальные шины соответственно передачи информации, адреса, управляющих сигналов и ключей защиты; СК1 — СК6 — селекторные каналы; МК — мультиплексорный канал

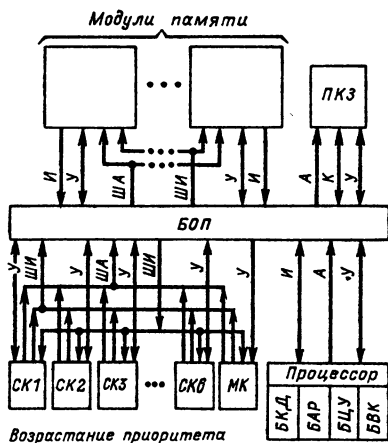
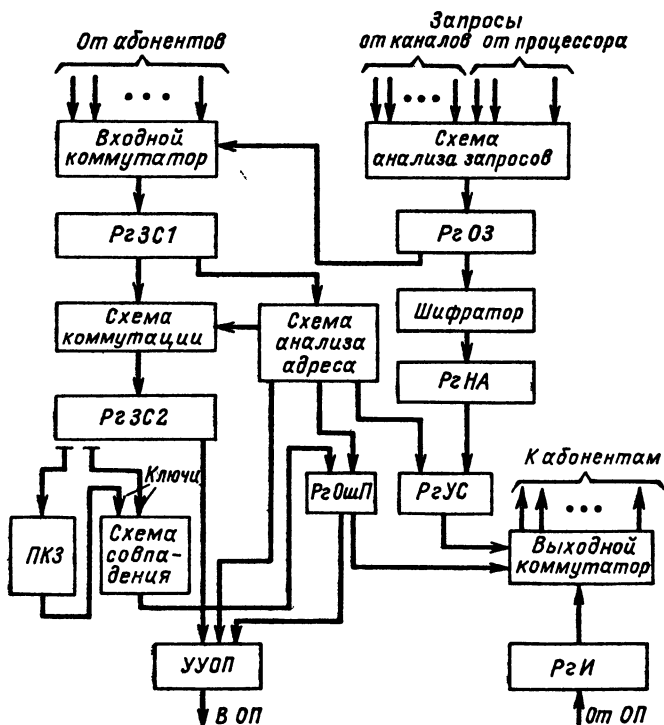


Рис. 14.6. Структурная схема блока обращений к оперативной памяти





На рис. 14.6 приведена упрощенная структурная схема БОП. Обработка запроса в БОП состоит из трех этапов. Поскольку на каждом этапе используются отдельные средства, возникает своеобразный конвейер, и при наличии запросов одновременно обрабатываются три запроса по одному для каждого этапа.

1. *Обработка сигналов запросов.* Схема анализа приоритетов запросов циклически с периодом, равным времени обработки одного запроса в ОП, анализирует поступившие от абонентов запросы, выделяет наиболее приоритетный из них и фиксирует его наличие засылкой 1 в соответствующий данному абоненту разряд регистра обобщенных запросов  $P_2OЗ$ .

Далее по содержимому  $P_2OЗ$  шифратор формирует в регистре номера адреса  $P_2НА$  4-разрядный номер абонента, принятого на обслуживание. При считывании информации из ОП номер абонента определяет, куда выдается информация. Код в  $P_2OЗ$ , управляя входным коммутатором, обеспечивает прием «запросного слова» (ЗС) соответствующего абонента в  $P_2ЗС1$ . Запросное слово содержит адрес ОП, признак обращения (запись или считывание), ключ программы, а при записи и информационное слово. На этом завершается первый этап.

2. *Обработка запросного слова.* Схема анализа адреса проверяет код адреса на нечетность, допустимость адреса для данной конфигурации ОП (нарушение адресации). Она также определяет по старшим разрядам  $P_2ЗС1$  номер модуля ОП, свободен ли этот модуль и формирует 7-разрядное управляющее слово. Это слово содержит 4-разрядный номер абонента, пославшего данный адрес, 2-разрядный номер модуля ОП и 1-разрядный признак операции обращения (запись или считывание).

Если адресуемый ЗС модуль памяти оказывается занятым операцией обращения для другого ЗС (в том числе от данного абонента), то ЗС остается в  $P_2ЗС1$  до освобождения модуля памяти, если ЗС принадлежит каналу, и освобождает  $P_2ЗС1$ , если принадлежит процессору. В последнем случае запросы процессора не обрабатываются до освобождения модуля памяти, оказавшегося занятым.

Если запрашиваемый модуль ОП свободен, ЗС передается в  $P_2ЗС2$  из  $P_2ЗС1$ , который, таким образом, освобождается для приема нового ЗС. На этом заканчивается второй этап.

3. *Обращение к модулю ОП.* В соответствии с содержимым управляющего слова запускается цикл обращения в определенном модуле ОП, записывается по указанному в ЗС адресу информационное слово из ЗС или производится считывание слова в  $P_2И$ . Далее считанная информация через выходной коммутатор, управляемый кодом номера абонента в  $P_2УС$ , передается соответствующему абоненту.

Одновременно с запуском цикла в ОП старшие разряды адреса (номер блока) передаются в качестве адреса в память ключей защиты. Происходит сопоставление ключа защиты из ПКЗ и ключа программы в ЗС. При их несоответствии в определенном разряде регистра ошибок памяти *РгОП* фиксируется нарушение защиты. В случае выявления нарушений при обращении к памяти (нарушение адресации или защиты), на что указывает наличие 1 в соответствующем разряде 3-разрядного регистра ошибки памяти *РгОшП*, блокируется запись слова из ЗС, а при считывании блокируется выдача считанного слова из *РгИ* абоненту (выдается нулевой код). Содержимое *РгОшП* выдается в регистр ошибок абонента.

Отметим в заключение, что с помощью БОП производятся запись ключей защиты в ПКЗ и считывание ключа при его передаче в регистр ССП.

Многоабонентное обслуживание порождает конфликты, возникающие при одновременном обращении абонентов к памяти. Эти конфликты, с которыми связаны простои отдельных абонентов в ожидании освобождения памяти, снижают производительность ВС. Для уменьшения частоты конфликтов используются быстродействующие буферные памяти в процессорах и каналах. Этой же цели служит расслоение памяти (см. § 14.2).

## **14.5. Динамическое распределение памяти.**

### **Организация виртуальной памяти**

В мультипрограммных системах размещение всех исполняемых программ полностью в ОП во многих случаях невыполнимо: программы часто имеют большую длину, а емкости существующих ОП ограничены. Однако нет принципиальной необходимости в том, чтобы вся программа находилась в ОП, так как в любой момент времени работа программы концентрируется на определенных сравнительно небольших участках. Таким образом, в ОП следует хранить только используемые в данный период части программ, а неиспользуемые части могут располагаться в ВЗУ. Программируя свою программу, пользователь не знает, в комбинации с какими программами будет выполняться его программа, какое место в памяти отведет ей операционная система.

При подготовке программ используются условные адреса. Позднее в процессе выполнения программы операционная система выделяет активным частям программы место в памяти, и условные адреса переводятся в исполнительные. Эта процедура получила название динамического распределения памяти.

Осуществление динамического распределения чисто программным путем привело бы к значительным потерям машинного времени. Целесообразнее пользоваться для этой цели аппаратными средствами.

Один из способов динамического распределения памяти основан на использовании базовых регистров. Операционная система каждой пользовательской программе ставит в соответствие свой базовый адрес. Базовые адреса обрабатываемых программ находятся в общих регистрах. При выполнении программы реальный или физический адрес образуется суммированием базового и относительного адресов. При динамическом распределении памяти с помощью базовых регистров программа (или, по крайней мере, та часть ее, адрес которой преобразуется с помощью одного и того же базового адреса) должна располагаться в последовательных ячейках и вводиться в ОП целиком, хотя в ближайшем цикле активности может потребоваться лишь небольшой фрагмент программы.

При рассматриваемом способе динамического распределения памяти свободная память может состоять из несвязанных областей (*фрагментация памяти*) и для ввода нужной программы может понадобиться сдвиг содержимого памяти. На рис. 14.7, а показано распределение памяти между программами *A, B, C, D*, из которых две (*A* и *D*) являются в данный момент наименее активными и следовательно, могут рассматриваться как кандидаты на удаление во внешнюю память. Если вновь вводимая программа *E* (рис. 14.7, б) больше любой из программ *A* и *D*, то для ее размещения в памяти необходимо, как показано на рис. 14.7, в, сдвигать программы *B* и *C*. Это перемещение связано с потерей времени. Более того, в ряде систем подобное перемещение требует выполнения заново операции редактирования связей в программе и новой загрузки программы.

Отмеченные недостатки в распределении памяти отсутствуют в виртуальной памяти со страничной организацией.

*Виртуальная память* есть способ организации памяти мультипрограммной вычислительной системы, при котором достигается гибкое динамическое распределение памяти, устраняется ее фрагментация и создаются значительные удобства для работы программистов. Это удастся достигнуть без заметного снижения

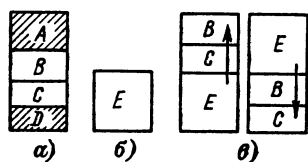


Рис. 14.7. Пример распределения памяти:

*а* — распределение памяти до ввода новой программы; *б* — вводимая программа; *в* — варианты распределения памяти после ввода новой программы

производительности машины ценой усложнения аппаратуры и операционной системы и процессов их функционирования.

Принцип виртуальной памяти предполагает, что пользователь при подготовке своей программы имеет дело не с физической ОП, действительно работающей в составе вычислительной установки и имеющей некоторую фиксированную емкость, а с виртуальной (т. е. кажущейся) одноуровневой памятью, емкость которой равна всему адресному пространству, определяемому размером адресных полей в форматах команд и базовых регистров. В ЕС ЭВМ это пространство составляет 16 777 216 байт.

Пользователь имеет в своем распоряжении все адресное пространство системы независимо от объема ее физической памяти и объемов памяти, необходимых для других программ, участвующих в мультипрограммной обработке.

На всех этапах подготовки программ, включая загрузку в оперативную память, программа представляется в *виртуальных адресах*, и лишь при самом исполнении машинной команды производится преобразование виртуальных адресов в реальные адреса действующей памяти (в так называемые *физические адреса*).

Преобразование виртуальных адресов в физические упрощается, и устраняется фрагментация памяти, если физическую и виртуальную память разбить на блоки, называемые в этом случае *страницами* и содержащие одно и то же число байт. Страницам виртуальной и физической памяти присваивают номера, называемые номерами соответственно виртуальных и физических страниц. Каждая физическая страница способна хранить одну из виртуальных страниц. Порядок расположения (нумерация) байт в виртуальной и физической страницах сохраняется одним и тем же.

В мультипрограммной системе страничная организация памяти дает определенные преимущества. Когда новая программа загружается в ОП, она может быть направлена в любые свободные в данный момент физические страницы независимо от того, расположены они подряд или нет. Не требуется перемещения информации в остальной части памяти. Страничная организация позволяет сократить объем передачи информации между внешней памятью и ОП, так как страница программы не должна загружаться до тех пор, пока она действительно не понадобится. Сначала в ОП загружается начальная страница программы, и ей передается управление. Если по ходу работы делается попытка выборки слов из другой страницы, то производится автоматическое обращение к операционной системе, которая осуществляет загрузку требуемой страницы.

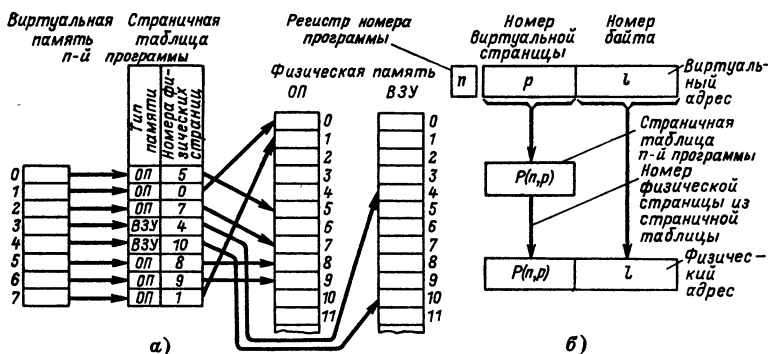


Рис. 14.8. Страничная организация памяти:  
 а — соответствие между виртуальными и физическими страницами; б — преобразование виртуального адреса в физический

На рис. 14.8 показано соответствие между виртуальной и физической памятью, устанавливаемое страничной таблицей, причем видно, что физические страницы могут содержаться в текущий момент времени как в оперативной, так и во внешней памяти. Страничная таблица для каждой программы формируется операционной системой в процессе распределения памяти и перерабатывается ею каждый раз, когда в распределении памяти производятся изменения. Процедура обращения к памяти состоит в том, что номер виртуальной страницы извлекается из адреса и используется для входа в страничную таблицу, которая указывает номер соответствующей физической страницы. Этот номер вместе с номером байта, взятым непосредственно из виртуального адреса, представляет собой физический адрес, по которому происходит обращение к ОП.

Если страничная таблица указывает на размещение требуемой информации во внешней памяти, то обращение к ОП не может состояться немедленно: операционная система должна организовать передачу из внешней памяти в ОП нужной страницы.

Для каждой из программ, обрабатываемых в мультипрограммном режиме, организуется своя виртуальная память и создается своя страничная таблица, при этом все программы делят между собой одну общую физическую память.

Страничные таблицы программ хранятся в ОП, и обращение к нужной строке активной страничной таблицы в ОП происходит *по адресу*, который определяется номером активной программы и номером виртуальной страницы (рис. 14.8).

Для ускорения преобразования адресов используется небольшая сверхоперативная память, куда передается из ОП стра-

ничная таблица активной программы. В другом варианте в сверхоперативной памяти могут находиться сведения о номерах виртуальных и соответствующих физических страниц для нескольких недавно использовавшихся страниц, в том числе принадлежащих разным программам. В этом варианте сверхоперативная память, используемая при преобразовании адресов, строится как *ассоциативная* (см. гл. 4) с обращением не адресным, а по содержанию хранимой в ячейке информации — в данном случае по хранимому в ячейке номеру программы к номеру виртуальной страницы.

*Сегментно-страничная организация памяти.* До сих пор предполагалось, что виртуальная память, которой располагает программист, представляет собой непрерывный массив с единой нумерацией байт. Однако программа обычно состоит из нескольких массивов — подпрограмм, одной или нескольких секций данных. Так как заранее длины этих массивов неизвестны, то удобно, чтобы при программировании каждый массив имел свою собственную нумерацию байт, начинающуюся с нуля и продолжающуюся в возрастающем порядке. Желательно также, чтобы составленная таким образом программа могла работать при динамическом распределении памяти, не требуя от программиста усилий по объединению различных ее частей в единый массив. Эта задача решается в некоторых вычислительных системах путем использования особого метода преобразования виртуальных адресов в физические, называемого *сегментно-страничной организацией памяти*.

Виртуальная память каждой программы делится на части, именуемые *сегментами*, с независимой адресацией байт внутри каждой части. К виртуальному адресу следует добавить дополнительные разряды левее номера страницы; эти разряды определяют номер сегмента.

Возникает определенная иерархия в организации программ, состоящая из четырех ступеней: 1) программа, 2) сегмент, 3) страница, 4) байт. Этой иерархии программ соответствует иерархия таблиц, служащих для перевода виртуальных адресов в физические. Программная таблица для каждой программы, загруженной в систему, указывает начальный адрес соответствующей сегментной таблицы. Сегментная таблица перечисляет сегменты данной программы с указанием начального адреса страничной таблицы, относящейся к данному сегменту. Страничная таблица определяет расположение каждой из страниц сегмента в памяти. Страницы сегмента могут располагаться не подряд, часть страниц данного сегмента может находиться в оперативной памяти, остальные — во внешней.

Рассмотрим с некоторыми упрощениями организацию сег-

ментно-страничной виртуальной памяти, реализованной в машинах ЕС ЭВМ II очереди [49].

В этой системе работа с виртуальной памятью возможна только в режиме расширенного управления (ССП [12] = 1).

Сегмент представляет собой блок последовательных адресов размером 64 Кбайт или 1 Мбайт, размер страницы 2 или 4 Кбайт. Начальные адреса сегментов и страниц кратны их размерам. Размеры сегментов и страниц виртуальной памяти активной в данный момент программы задаются значениями соответствующих разрядов управляющего регистра 0.

Виртуальный адрес (как и физический) имеет длину 24 разряда, причем поле номера сегмента занимает 8 или 4 старших разряда соответственно для сегментов размером 64 Кбайт и 1 Мбайт, поле номера байта занимает 11 или 12 младших разрядов для страниц размером 2048 и 4096 байт. Промежуточные разряды адреса занимает поле номера страниц, которое может иметь 4, 5, 8 или 9 разрядов в зависимости от размеров сегмента и страницы.

Сегментные и страничные таблицы находятся в ОП, а в программной таблице нет необходимости, так как для каждой активной в данный момент программы управляющий регистр 1 хранит начальный адрес и длину соответствующей сегментной таблицы. Хранит он также номер программы.

Процесс преобразования адресов представлен на рис. 14.9. В общем случае преобразование адреса происходит в два этапа и требует двух дополнительных обращений к ОП (рис. 14.9, а).

**Первый этап.** Начальный адрес сегментной таблицы, установленный в управляющем регистре 1, суммируется с номером сегмента из виртуального адреса. В результате образуется адрес, по которому из ОП считывается строка сегментной таблицы, содержащая адрес начала и длину страничной таблицы для данного сегмента.

**Второй этап.** Полученный адрес начала страничной таблицы суммируется с номером страницы из виртуального адреса, при этом образуется адрес, по которому из ОП считывается строка страничной таблицы. Если эта страница оказывается в ОП, то в старшие разряды регистра физического адреса передается ее номер, а в младшие заносится номер байта из регистра виртуального адреса. Формирование физического адреса на этом завершается.

Если нужная физическая страница оказывается во внешней памяти, то происходит прерывание по страничному сбою (см. § 14.6). Операционная система инициирует передачу этой страницы из внешней памяти в ОП (при этом меняется номер физической страницы) и корректирует соответствующим обра-

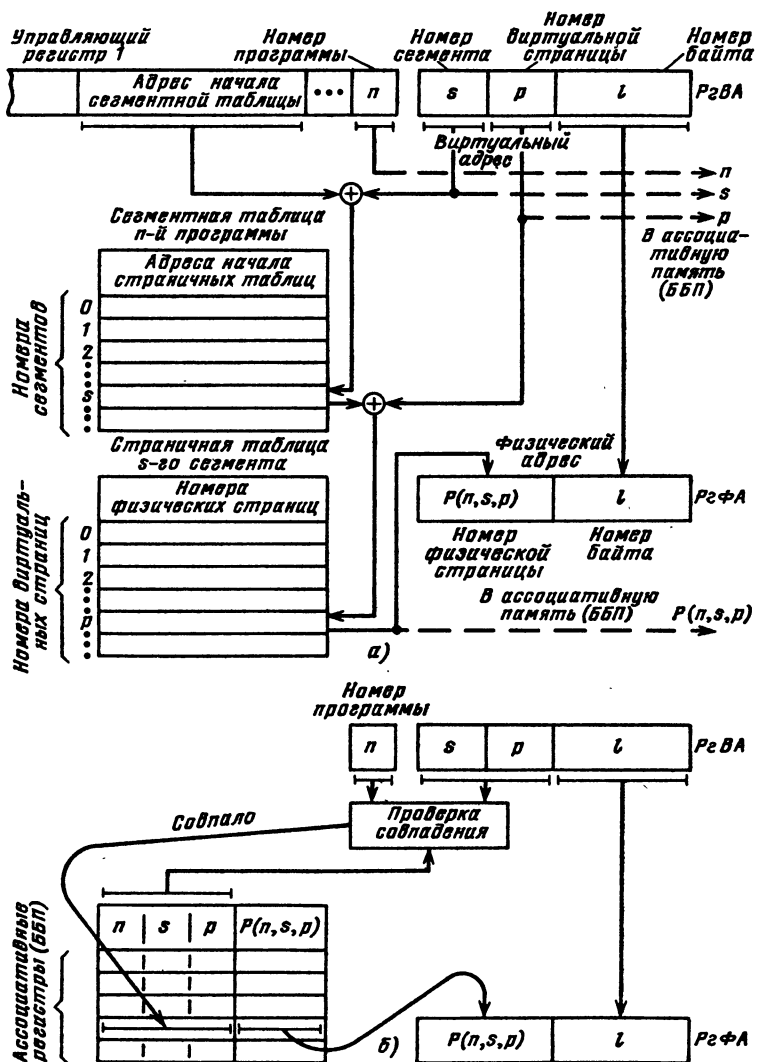


Рис. 14.9. Преобразование адресов при сегментно-страничной организации памяти:

*а* — с использованием таблиц в ОП; *б* — с использованием ассоциативной памяти (блока быстрой переадресации)



зом страничную таблицу, находящуюся в оперативной памяти. В старшие разряды регистра физического адреса передается новый номер физической страницы, а в младшие — номер байта. Физический адрес сформирован.

Выполняется запрошенное программой обращение к ОП. Одновременно информация о текущей странице (номерах программы, сегмента, виртуальной и соответствующей физической страницы) помещается в сверхоперативную ассоциативную память [или в блок быстрой переадресации (ББП)] небольшой емкости.

Ассоциативная память (ББП) хранит указанные данные для небольшого числа недавно использовавшихся страниц. Так, в ЭВМ ЕС-1045 ББП представляет собой память на интегральных микросхемах емкостью 128 слов с временем выборки 25—30 нс [67]. Блок быстрой переадресации может обслуживать одновременно до 3 программ, другими словами, до трех виртуальных памяти.

При наличии ассоциативной памяти (ББП) значительно ускоряется процесс преобразования адресов, так как на каждом участке вычислительного процесса обращения к ОП сосредотачиваются на небольшом числе страниц, и поэтому имеется большая вероятность, что текущее обращение произойдет к странице, информация о которой уже имеется в ассоциативной памяти (ББП), а следовательно, возможно быстрое преобразование адресов без дополнительных обращений к ОП.

Преобразование адресов всегда начинается с просмотра ассоциативной памяти (ББП), и если оказывается, что в одной из ее строк (ассоциативном регистре) хранится информация о странице, к которой должно произойти обращение, то из этой строки непосредственно выбирается номер физической страницы и дополнительные обращения к ОП (к сегментной и страничной таблицам) не производятся (рис. 14.9, б).

Если нужной информации нет в ассоциативной памяти (ББП), то делается попытка сократить время преобразования путем исключения одного дополнительного обращения к ОП (первый этап на рис. 14.9, а). Может оказаться, что страница, к которой происходит обращение, принадлежит сегменту предыдущего обращения к ОП. В аппаратуре преобразования адресов сохраняются номер сегмента и адрес начала его страничной таблицы для предыдущего обращения. Если совпадают номера сегментов текущего и предыдущего обращений, первый этап преобразования исключается, используется сохраненный адрес начала сегментной таблицы и выполняется только второй этап преобразования, т. е. производится только одно дополнительное обращение к ОП. Если номера сегментов не совпадут, реализуется

ся полная процедура преобразования адресов, показанная на рис. 14.9, а.

Дополнительные обращения к ОП сопровождаются занесением информации о текущей странице в ассоциативную память (ББП). Если в ассоциативной памяти (ББП) не оказывается свободного регистра (строки), данные о новой странице записываются на место данных, которые дольше других не использовались в процессе преобразования адресов.

#### 14.6. Алгоритмы управления многоуровневой памятью

Будем рассматривать двухуровневую память со страничной организацией, состоящую из оперативной (верхний уровень) и внешней (нижний уровень) памяти.

Если при выполнении программы обнаруживается, что страница с нужными данными (операндами, куском программы) отсутствует в памяти верхнего уровня, она передается туда из памяти нижнего уровня. Если при этом в памяти верхнего уровня нет для нее свободного места, то она замещает одну из страниц, находящихся в ОП. При этом, если в замещаемую страницу во время ее пребывания в ОП производилась запись, она должна быть передана в память нижнего уровня (заменит там свою устаревшую копию).

Эти операции передачи информации между уровнями памяти вызывают простои процессора и, следовательно, потери производительности вычислительной системы. Поэтому следует стремиться уменьшить число таких операций в процессе выполнения программы. Очевидно, что число этих операций обмена информацией зависит от того, какая информация отсылается из памяти верхнего уровня, так как к этой информации возможно обращение в процессе дальнейшего выполнения программы.

Приведем формализованную модель процесса обмена информацией между верхним и нижним уровнями памяти [10].

Пусть программа вместе с исходными данными состоит из  $k$  страниц, которым присвоены номера 1, 2, ...,  $k$ , тогда программу можно рассматривать как множество страниц

$$Q = \{1, 2, \dots, k\}.$$

Все страницы программы постоянно хранятся в памяти нижнего уровня, а кроме того,  $r$  из них могут находиться в памяти верхнего уровня (оперативной памяти), при этом

$$1 < r < k.$$

Выполнение программы порождает последовательность обращений к страницам памяти. Рассматриваем эту последовательность как реализацию некоторого случайного процесса

$$q_0, q_1, q_2, \dots, q_t,$$

где  $q_t$  — случайная дискретная величина, принимающая в момент времени  $t$  значение одного из номеров страниц программы ( $q_t \in Q$ ).

Если  $S_t$  — совокупность страниц в памяти верхнего уровня в момент  $t$ , причем в любой момент в этой памяти присутствует  $r$  страниц программы, то изменение состояния памяти верхнего уровня после обращения  $q_t$  описывается следующими соотношениями:

$$S_{t+1} = \begin{cases} S_t, & \text{если } q_t \in S_t; \\ S_t - v_t + q_t, & \text{если } q_t \notin S_t. \end{cases}$$

В первом случае обращение производится к странице, которая находится в памяти верхнего уровня, и поэтому состояние этой памяти не меняется.

Во втором случае происходит обращение к странице, отсутствующей в памяти верхнего уровня. Эта ситуация называется *страничным сбоем*, так как программа не может дальше выполняться, пока нужная страница  $q_t$  не будет переписана из памяти нижнего уровня в память верхнего уровня, что сопряжено с потерями времени. Поскольку в памяти верхнего уровня нет свободного места, из нее приходится удалять некоторую страницу  $v_t$ , с тем чтобы на ее место можно было поместить страницу  $q_t$ . Если во время пребывания страницы  $v_t$  в памяти верхнего уровня в нее производилась запись, эта страница при замещении должна переписываться в память нижнего уровня. Такая процедура называется *процессом замещения страниц*, а правило, по которому при возникновении страничного сбоя выбирается страница  $v_t \in S_t$  для удаления из памяти верхнего уровня, — *алгоритмом замещения*.

Для данной программы, порождающей некоторый поток обращений к памяти, существует по крайней мере одна такая последовательность замещений страниц, которая дает минимальное для этой программы число страничных сбоев — *минимально возможную последовательность замещений*. При конструировании алгоритма замещений стремятся приблизить реализуемую этим алгоритмом последовательность замещений к минимальной.

Оптимизация процесса замещений страниц упрощается, если известно, в каком порядке в будущем будут происходить обращения к памяти или, по крайней мере, вероятности обращений в будущем к отдельным страницам программы. Ясно, например,

что в первую очередь из памяти верхнего уровня следует удалить страницу, к которой обращений больше не будет (вероятность обращений в будущем равна 0).

Трудность состоит в том, что, как правило, при выполнении программы отсутствуют информация о потоке обращений или сколько-нибудь достоверные сведения о вероятности обращений к отдельным страницам в будущие моменты времени.

Алгоритмы замещения можно разделить на две группы:

1) *физически нереализуемые*, использующие информацию (реально отсутствующую) о потоке обращений в будущие моменты времени;

2) *физически реализуемые* или эвристические, использующие только информацию об обращениях к памяти в прошедшие моменты времени, т. е. только историю процесса.

Хотя алгоритмы первой группы на практике применить нельзя, они играют важную роль в теории алгоритмов замещения, позволяя производить оценки (в том числе экспериментальные), в какой степени характеристики эвристических алгоритмов приближаются к предельно возможным оптимальным.

*Физически нереализуемые алгоритмы. Алгоритм Михновского — Шора*<sup>1</sup>. При каждом замещении страницы из памяти верхнего уровня отсылается в память нижнего уровня страница, очередное обращение к которой произойдет позже, чем к любой другой странице в памяти верхнего уровня.

Справедливо следующее предложение. Число замещений страниц в памяти верхнего уровня (число страничных сбоев) при выполнении замещений по алгоритму Михновского — Шора является минимальным для заданных потока обращений и исходного распределения памяти.

Доказательство справедливости предложения можно найти в [39].

Таким образом, алгоритм Михновского — Шора реализует минимально возможную для данной программы последовательность замещений. Поэтому этот алгоритм называют *МИН-алгоритмом*.

Если условиться, что известна вероятность обращений к отдельным страницам программы, то оптимальным в смысле минимума среднего числа страничных сбоев является *ОПТ-алгоритм*: при каждом замещении страницы из памяти верхнего уровня

---

<sup>1</sup> Алгоритм и доказательство его оптимальности впервые опубликованы С. Д. Михневским и Н. З. Шором в 1965 г. [39]. В литературе часто этот алгоритм приписывают американскому автору Беледи на основании его статьи, опубликованной в 1966 г.

отсылается страница, вероятность обращения к которой не больше, чем к любой другой странице в этой памяти.

*Физически реализуемые (эвристические) алгоритмы замещения.* Был предложен ряд алгоритмов этого класса.

*Алгоритм случайного замещения (СЗ-алгоритм).* При возникновении страничного сбоя из памяти верхнего уровня с равной вероятностью отсылается любая из находящихся там страниц.

*НДИ-алгоритм.* Из памяти верхнего уровня отсылается страница, наиболее давно использовавшаяся.

*Алгоритм «первый пришел — первый ушел» (ПППУ-алгоритм).* Отсылается страница, дольше других находившаяся в памяти верхнего уровня.

*Алгоритм «последний пришел — первый ушел».* Отсылается страница, позже других поступившая в память верхнего уровня.

Следующие два алгоритма обладают определенными свойствами адаптации к потоку обращений к памяти.

*Алгоритм «карабкающаяся страница» (КС-алгоритм).* Страницы в памяти верхнего уровня образуют последовательность (рис. 14.10)

$$S_t = j_1, j_2, \dots, j_{m-1}, j_m, j_{m+1}, \dots, j_r,$$

которая при очередном обращении  $q_t$  к памяти изменяется по правилу

$$S_{t+1} = \begin{cases} S_t & \text{при } q_t = j_1; \\ j_1, j_2, \dots, j_m, j_{m-1}, \dots, j_r & \text{при } q_t = j_m, m \neq 1; \\ j_1, j_2, \dots, j_{r-1}, q_t & \text{при } q_t \notin S_t. \end{cases}$$

При обращении к странице  $j_m$ , присутствующей в памяти верхнего уровня, последняя меняется местами с соседней слева страницей, другими словами, «карабкается» к началу последова-

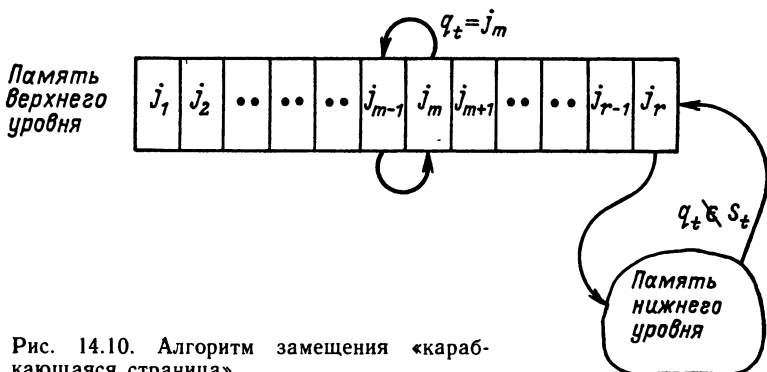


Рис. 14.10. Алгоритм замещения «карабкающаяся страница»

тельности, подальше от ее конца, куда происходит замещение при страничном сбое.

*Алгоритм «рабочий комплект» (РК-алгоритм).* Страницы в памяти верхнего уровня, использовавшиеся в течение заданного интервала времени, образуют «рабочий комплект». Страницы из этой памяти, не вошедшие в «рабочий комплект», формируют две очереди кандидатов на замещение: 1) очередь страниц, в которые не вносились изменения, пока они присутствовали в памяти верхнего уровня; 2) очередь страниц, в которые вносились изменения. Замещение при страничном сбое производится по правилу: первый пришел из рабочего комплекта — первый ушел из памяти верхнего уровня, при этом сначала подлежат замещению страницы из первой очереди.

Формирование указанных очередей производится с использованием ключа защиты, расширенного дополнительными разрядами *Об* и *Из* (см. рис. 14.4), устанавливаемыми в 1, если соответственно происходит любое обращение к данному блоку (странице) и если в него произведена запись. Описанный алгоритм применен в ЕС ЭВМ.

Предположим, что последовательность обращений  $q_1, q_2, \dots, q_t$  соответствует последовательности независимых случайных дискретных величин, таких, что

$$P\{q_t = j\} = p_j, \quad 1 \leq j \leq k, \quad \sum_{j=1}^k p_j = 1.$$

Примем за состояние процесса замещения набор ( $a$  в некоторых случаях упорядоченную последовательность) страниц, находящихся в памяти верхнего уровня. Тогда для ряда алгоритмов замещения (СЗ, НДИ, ПППУ и некоторых других) процесс изменения состояния верхнего уровня описывается однородной конечной эргодической цепью Маркова, что указывает на существование стационарных вероятностей пребывания процесса в определенных состояниях и, как следствие этого, стационарных вероятностей страничных сбоев.

В качестве критерия эффективности  $W_{r,k}$  алгоритма замещения  $A$  примем стационарную вероятность страничных сбоев

$$W_{r,k}(A) = \lim_{t \rightarrow \infty} P\{q_t \notin S_t\}.$$

Можно для ряда алгоритмов замещения найти зависимость  $W_{r,k}$  от  $p_1, p_2, \dots, p_k$  и сравнить алгоритмы между собой и также с физически нереализуемым ОПТ-алгоритмом. Определить  $W_{r,k}$  для ряда алгоритмов можно, использовав метод, основанный на однородных эргодических цепях Маркова [10].

## 14.7. Тенденции развития ЭВМ общего назначения

Быстрое развитие микропроцессорных средств и форсированное улучшение характеристик основанных на их использовании персональных компьютеров, микроЭВМ, супермини- и супермикроЭВМ заставляет задуматься о будущем машин общего назначения и возможных тенденциях их развития.

Обусловленные в значительной степени универсальностью назначения сложность программного обеспечения основных моделей ЕС ЭВМ, их громоздкость, сложность технического обслуживания стимулируют переориентацию пользователя (если его задачи это позволяют) на использование профессиональных персональных компьютеров, мини- и микроЭВМ.

Однако непрерывное усложнение подлежащих решению задач, возрастание требований к производительности машин, точности вычислений, объемам хранимой и обрабатываемой информации диктуют необходимость развития ЭВМ общего назначения в направлении создания предназначенных для широкого использования мощных ЭВМ общего назначения (*суперЭВМ общего назначения*) с производительностью 50—100 млн. операций/с, реализующих эффективные режимы общения с пользователем, обладающих многочисленным и разнообразным периферийным оборудованием и системой ввода-вывода большой пропускной способности. Эти машины должны иметь повышенную надежность и живучесть, содержать в своем составе эффективные средства поддержки эксплуатационного обслуживания.

Имеется существенное обстоятельство, которое, с одной стороны, поддерживает ориентацию пользователя на машины общего назначения, а другой — вносит ограничения на решения, направленные на их развитие.

Этим обстоятельством является огромное накопленное в стране программное обеспечение для этих машин, оцениваемое суммой в несколько миллиардов рублей. Поэтому при развитии машин общего назначения ЕС ЭВМ приходится учитывать необходимость сохранения возможности использования имеющегося программного обеспечения, т. е. необходимость обеспечивать архитектурную, в том числе программную совместимость новых моделей с предыдущими моделями этих машин.

В архитектуре машин общего назначения достигнут высокий уровень международной унификации, который в дальнейшем следует сохранить, так как это создает условия для международного сотрудничества в создании программных продуктов.

Сказанное диктует, по меньшей мере на ближайший период, эволюционный путь развития архитектуры машин общего на-

значения, но не ставит ограничений в отношении самой технологии реализации их архитектурных свойств.

При значительном росте производительности затрудняется сбалансирование пропускных способностей процессорной части, памяти, системы ввода-вывода и самих внешних ЗУ. Наряду с увеличением быстродействия потребуется значительное увеличение емкости физической ОП и виртуальной памяти, а следовательно, и разрядности адреса. Увеличение длины адреса, например, до 31 разряда позволит увеличить виртуальное адресное пространство машины до  $2^{31}$  байт (2 Гбайт). В структуре памяти может появиться больше уровней с увеличением размера информационных единиц, используемых при обменах между промежуточными уровнями ОП. Должны увеличиться емкость и быстродействие различных буферных памяти.

Для увеличения пропускной способности системы ввода-вывода придется увеличить число каналов ввода-вывода до нескольких десятков. Для полного освобождения средств процессора от управления операциями ввода-вывода и от использования его аппаратуры в рабочих процедурах каналов понадобится создать функционально развитые процессоры ввода-вывода («директора каналов»).

Для достижения указанной выше производительности следует продолжать уменьшать продолжительность такта процессора, применяя более быстродействующие БИС, искать конструктивные решения по отводу тепла от электронной аппаратуры. Но одно это не решит проблему повышения производительности машины. Будущие ЭВМ общего назначения будут представлять собой вычислительные системы, содержащие несколько взаимодействующих процессоров универсального и специализированного (для определенных типов вычислений) назначений.

В процессорах будет широко использоваться конвейеризация обработки команд и операций над операндами. Их аппаратура будет включать в себя средства для выполнения векторных операций, а система команд пополнится векторными командами.

ЭВМ общего назначения приобретут много черт, которые недавно приписывались лишь вычислительным системам сверхвысокой производительности (см. гл. 15).

Развитие ЭВМ общего назначения, конечно, не может пройти мимо существующей тенденции *интеллектуализации средств вычислительной техники*, которая связана с реализацией средствами ЭВМ все более сложных функций и процессов обработки информации и существенного упрощения общения человека с машиной при подготовке и решении задач.

Все в большей степени функции операционных систем будут «погружаться» в аппаратуру, машины будут снабжаться про-



граммно-аппаратурными средствами поддержки крупных баз данных, организации баз знаний, реализации на их основе крупных экспертных систем.

Что касается облегчения общения человека с машиной, то реализация таких средств и функций, как речевой ввод и вывод, системы общения на естественном языке, обучающие и поддерживающие работу пользователя на машине, по-видимому, будет возложена на интеллектуальные терминалы машин общего назначения, построенные на основе персональных компьютеров.

В качестве иллюстрации к высказанным выше соображениям рассмотрим новую модель машины общего назначения — систему 3090/400 фирмы IBM, являющейся в мировой технике ведущей в области машин рассматриваемого класса [86].

В машине IBM 3090 использованы или получили развитие некоторые схемные и архитектурные решения, впервые осуществленные в непосредственно ей предшествующих моделях ЭВМ фирмы IBM. Так, организация каналов ввода-вывода заимствована из ЭВМ IBM 370 XA<sup>1</sup> [70], организация векторных операций является развитием векторных средств, использованных в IBM 370 XA и IBM 3033, в организации кэш-памяти критический осмыслен опыт, полученный при разработке IBM 3080.

*Схемотехника ЭВМ IBM 3090* основана на применении специализированных БИС, содержащих до 704 вентиляционных схем ЭСЛ. По сравнению с использовавшимися в IBM 3080 БИС примерно с подобным же числом схем ТТЛ БИС IBM 3090 не только обладают большим быстродействием, но и, что важно, имеют парафазные сигналы на выходах, позволяющие отказаться от дополнительных схем инвертирования сигналов. На этих же кристаллах удастся разместить умощнители выходных сигналов, для которых в IBM 3080 требовались отдельные интегральные микросхемы.

В результате схемотехника IBM 3090 позволила снизить продолжительность такта до 18,5 нс, в то время как в IBM 3080 она составляла 26—24 нс, а в IBM 3033, построенной на фиксированном наборе кристаллов, достигала 57 нс.

Увеличение степени интеграции и компактности оборудования машины достигнуто конструкторскими решениями, основанными на использовании теплопроводящих модулей (ТСМ), содержащих до 100 БИС. В свою очередь, модули по 6 и 9 шт. располагаются на панелях. Таким образом, исключается существовавший в предыдущих моделях ЭВМ платовый (ТЭЗовый) уровень конструкции. Значительные трудности представляет отвод тепла от электронной аппаратуры. Модуль IBM 3090 потребляет около 500 Вт. В теплопроводящем модуле это решается применением прилегающей к корпусам кристаллов наполненной гелием металлической камеры в сочетании с водяным охлаждением.

---

<sup>1</sup> XA — расширенная архитектура (от Extended Architecture).

Общее число электронных схем в IBM 3090 достигает 300 тыс. Разработка такой машины потребовала использования мощной системы автоматизации проектирования, включая систему потактового моделирования.

Структура машины IBM 3090/400 представлена на рис. 14.11. Модель 3090/400 состоит из двух двухпроцессорных машин IBM 3090/200, имеющих каждая свою центральную (оперативную) и расширенную памяти, свою каналную систему.

Взаимодействием основных устройств каждой из машин 3090/200 управляют их блоки управления системой (БУС). Связь между машинами 3090/200 осуществляется по шинам, соединяющим их блоки управления системой и их расширенные памяти.

Новый эффективный структурный элемент — расширенная память емкостью несколько сотен мегабайт и более — не является дополнительной центральной памятью. Ее основное назначение — служить буферной памятью между ОП и дисковыми ЗУ для компенсации большого времени доступа и сравнительно низкой скорости передачи данных в этих устройствах, могущих вызвать значительные простои в работе процессора. Благодаря расширенной памяти пропускная способность внешних ЗУ увеличивается (виртуально по отношению к процессору) в несколько сотен раз. В расширенной памяти адресуемой и участвующей под управлением операционной системы в обмене с центральной памятью единиц информации является 4-килобайтная страница. Это облегчает адресацию памяти большой емкости.

В структуре центрального процессора (рис. 14.12), как это принято в современных процессорах и микропроцессорах, выделены командный

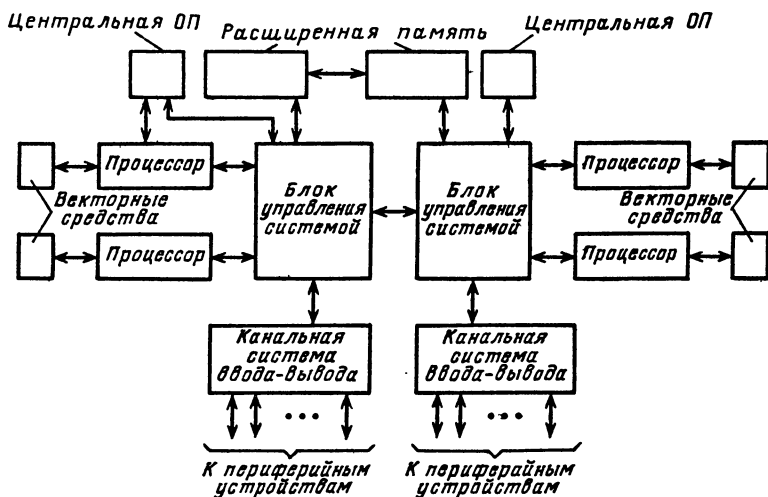


Рис. 14.11. Структура ЭВМ общего назначения IBM 3090

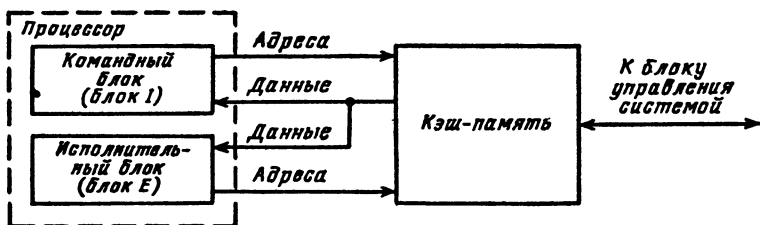


Рис. 14.12. Структура процессора ЭВМ IBM 3090

блок (блок *I*) и исполнительный (блок *E*). Эти блоки связаны друг с другом и со скрытой (недоступной программисту) кэш-памятью. Блоки *I* и *E* выполнены с использованием конвейерной обработки и перекрытия, так что в рассматриваемом процессоре реализован конвейер команд (одновременная обработка четырех команд) и арифметический конвейер. В блоке *E* применено микропрограммное управляющее устройство (длина микрокоманды свыше 100 разрядов) с управляющей памятью, допускающее считывание и с меньшей скоростью запись.

В блоке *I* используется буфер для предварительной выборки команд, хранения очереди четырех декодированных команд вместе с их операндами и согласования скоростей работы блоков *I* и *E*.

Средства векторной обработки каждого центрального процессора включают в себя конвейерное арифметическое устройство и набор из 16 векторных регистров по 128 32-битных элементов в каждом. Эти средства реализуют 171 векторную команду. Они являются дополнением к набору команд системы IBM 370. При заполненности конвейера время обработки пары операндов (элементов двух векторов) близко к одному машинному такту, причем, чтобы снизить время операции умножения до одного такта, применен подконвейер из трех множительных устройств, поочередно загружаемых парами перемножаемых элементов векторов.

Кэш-память в IBM 3090 представляет собой быстродействующую память-буфер емкостью 64 Кбайт с временем обращения 2 машинных такта. В системе IBM 3090 в целях повышения быстродействия отказались от одновременной записи информации в кэш- и основную память (так называемой кэш типа *store-through*). Новая информация записывается только в кэш (кэш типа *store in cache*), при этом исключается цикл более медленной центральной памяти, но возникает проблема выравнивания (актуализации) содержания центральной памяти. «Строка» (единица информации размером в четыре двойных слова, которыми обмениваются кэш-память и центральная память) передается из кэш-памяти в центральную память только в том случае, если ее запрашивает другой процессор или надо освободить место в кэш-памяти. С привлечением второго БУС возможен прямой обмен информацией между кэш-памятью-

ми процессоров, принадлежащих разным ЭВМ модели 3090/200, объединенным в систему 3090/400.

*Канальная система.* Каждая из машин, входящих в состав IBM 3090/400, имеет собственную канальную систему. Необходимость приведения в соответствие пропускной способности системы ввода-вывода со значительно увеличившимися скоростью работы процессорной части и пропускной способностью центральной памяти (в условиях сравнительно медленного увеличения скорости передачи данных у внешних ЗУ и других ПУ) потребовала увеличения числа каналов ввода-вывода. Так, в каждой из двух канальных систем IBM 3090/400 число каналов может достигать 48.

При увеличении числа каналов значительно возрастает объем операций по управлению вводом-выводом, в результате чего особую остроту приобретает проблема более полного освобождения процессора от процедур, связанных с вводом-выводом. Тот уровень освобождения процессора от этих процедур, который достигнут в выпускаемых в настоящее время машинах общего назначения (см. гл. 11), оказывается в данном случае недостаточным. Там процессор освобождается главным образом от управления реализацией канальных программ, что, конечно, весьма существенно. Однако во многих случаях часть аппаратуры процессора, например управляющая память, используется процессором и каналами на основе разделения времени, что снижает производительность процессора. Кроме того, инициализация любой операции ввода-вывода сопровождается переходом ЭВМ в режим супервизора (по команде «Вызов супервизора») и выполнением процессором программы «супервизор ввода-вывода».

В новой системе ввода-вывода IBM 3090 достигается существенное освобождение процессора от процедур, связанных с операциями ввода-вывода. На рис. 14.13, заимствованном из [70], сопоставляются (применительно к двухпроцессорным комплексам) структуры существенно различающихся существующей (а) и новой (б) систем ввода-вывода.

В новой, «канальной системе» процедуры, связанные с выполнением канальных программ и управлением интерфейсом ввода-вывода, выполняют каналы, представляющие собой микропрограммируемые процессоры с управляющей памятью, допускающей запись микропрограмм для задания нужной комбинации байт- и блок-мультиплексных каналов и их режимов работы. Вся канальная система управляется микропрограммируемым процессором ввода-вывода («директором каналов») с сокращенным набором команд [с RISC-архитектурой (см. гл. 9)], которые при инициализации процессором операции ввода-вывода выполняют своими микропрограммными средствами функции «супервизора ввода-вывода», организуют очереди запросов ввода-вывода, управляют выбором маршрутов передачи информации в системе ввода-вывода (в условиях многовариантности этих маршрутов), формируют запросы прерывания, осуществляют динамическое повторение операции ввода-вывода при ошиб-

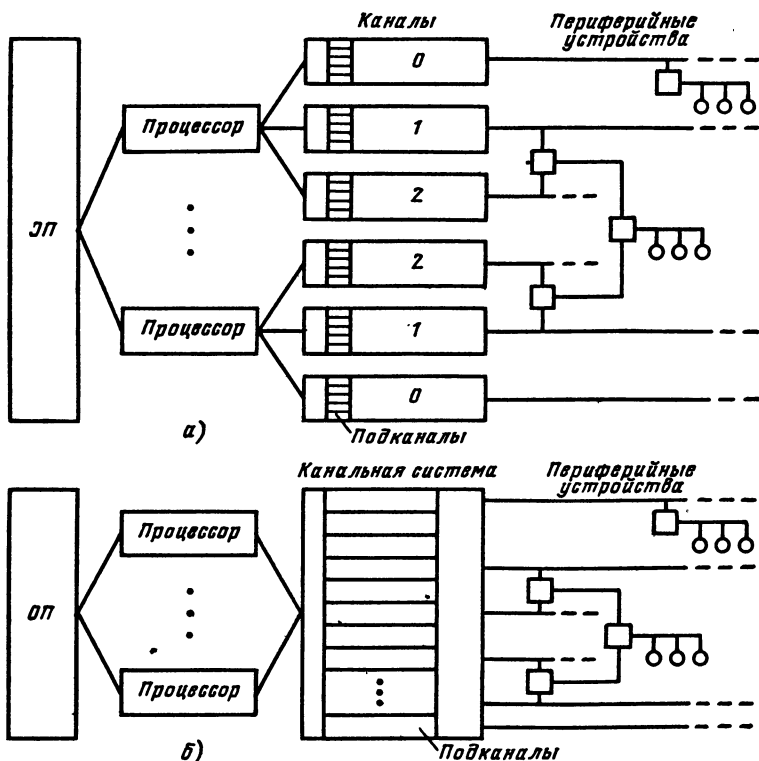


Рис. 14.13. Сопоставление структур систем ввода-вывода:  
 а — существующая система ввода-вывода машин общего назначения; б — «канальная система» в IBM 3090

ках. В канальной системе используется логическая адресация периферийных устройств, не зависящая от маршрута передачи данных. Сами маршруты могут меняться на разных этапах выполнения одной и той же цепочки операций.

В рассматриваемой системе каждому процессору доступны любой канал и любое периферийное устройство; другими словами, реализуются «общее поле каналов ввода-вывода» и «общее поле периферийных устройств» (см. гл. 15).

### Контрольные вопросы

1. Для чего предназначена кэш-память? Каким образом автоматически устанавливается наличие запрашиваемой процессором информации в кэш-памяти?

2. Какие существуют режимы защиты и в чем их различие? Почему время обращения к памяти ключей защиты должно быть существенно меньше времени обращения к ОП?

3. Поясните, как в схеме на рис. 14.6 реализуется конвейерная обработка запросов на доступ к ОП?

4. Каким образом сегментно-страничная виртуальная память способствует более эффективному использованию ОП и облегчает труд программистов?

5. Почему в устройстве быстрого преобразования адресов оказалось необходимым использование ассоциативной памяти?

6. В чем различие физически реализуемых и нереализуемых алгоритмов замещения страниц? Как можно использовать физически нереализуемые алгоритмы замещения страниц?

## Глава 15

# **ПРИНЦИПЫ ОРГАНИЗАЦИИ МНОГОПРОЦЕССОРНЫХ И МНОГОМАШИННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ (КОМПЛЕКСОВ) И СУПЕРЭВМ**

## **15.1. Понятие о многомашинных и многопроцессорных вычислительных системах и комплексах**

Вычислительная техника в своем развитии по пути повышения быстродействия ЭВМ приблизилась к физическим пределам. Время переключения электронных схем достигло долей наносекунды, а скорость распространения сигналов в линиях, связывающих элементы и узлы машины, ограничена значением 30 см/нс (скорость света). Поэтому дальнейшее уменьшение времени переключения электронных схем не позволит существенно повысить производительность ЭВМ. В этих условиях требования практики (сложные физико-технические расчеты, автоматизированное проектирование сложных объектов, многомерные экономико-математические модели и другие задачи) по дальнейшему повышению быстродействия ЭВМ могут быть удовлетворены только путем распространения принципа параллелизма на сами устройства обработки информации и создания *многомашинных и многопроцессорных (мультипроцессорных) вычислительных систем*. Такие системы позволяют производить распараллеливание во времени выполнения программы или параллельное выполнение нескольких программ.

В настоящее время исключительно важное значение приобрела проблема обеспечения высокой надежности и готовности вычислительных систем, работающих в составе различных автоматизированных систем обработки данных и управления, особенно при работе в режиме реального времени. Эта проблема решается на основе использования *принципа избыточности*, который ориентирует также на построение многомашинных или многопроцессорных систем (комплексов). Появление дешевых и небольших по размерам микропроцессоров и микроЭВМ облегчило построение и расширило область применения многопроцессорных и многомашинных ВС разного назначения.

Различие понятий многомашинной и многопроцессорной ВС поясняет рис. 15.1. Многомашинная ВС (ММС) содержит несколько ЭВМ, каждая из которых имеет свою ОП и работает под управлением своей операционной системы, а также средства обмена информацией между машинами. Реализация обмена информацией происходит в конечном счете путем взаимодействия операционных систем машин между собой. Это ухудшает динамические характеристики процессов межмашинного обмена данными. Применение многомашинных систем позволяет повысить надежность вычислительных установок. При отказе в одной машине обработку данных может продолжать другая машина. Однако можно заметить, что при этом оборудование комплекса недостаточно эффективно используется для этой цели. Достаточно в системе, изображенной на рис. 15.1, а, в каждой ЭВМ выйти из строя по одному устройству (даже разных типов), как вся система становится неработоспособной.

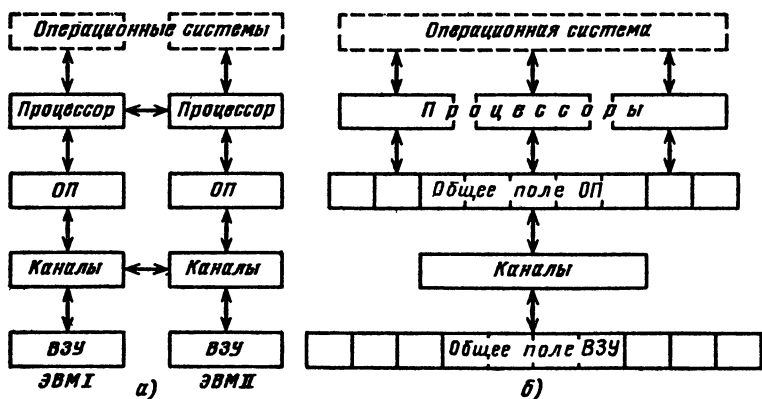


Рис. 15.1. К сопоставлению многомашинной (а) и многопроцессорной (б) систем

Этих недостатков лишены многопроцессорные системы (МПС). В таких системах (рис. 15.1, б) процессоры обретают статус рядовых агрегатов вычислительной системы, которые подобно другим агрегатам, таким, как модули памяти, каналы, периферийные устройства, включаются в состав системы в нужном количестве.

Вычислительная система называется многопроцессорной, если она содержит несколько процессоров, работающих с общей ОП (*общее поле оперативной памяти*), и управляется одной общей операционной системой. Часто в МПС организуется общее поле внешней памяти.

Под общим полем понимается равнодоступность устройств. Так, общее поле памяти означает, что все модули ОП доступны всем процессорам и каналам ввода-вывода (или всем периферийным устройствам в случае наличия общего интерфейса); общее поле ВЗУ означает, что образующие его устройства доступны любому процессору и каналу.

В МПС по сравнению с ММС достигается более быстрый обмен информацией между процессорами (через общую ОП), и поэтому может быть получена более высокая производительность, более быстрая реакция на ситуации, возникающие внутри системы и в ее внешней среде, и более высокие надежность и живучесть, так как система сохраняет работоспособность, пока работоспособны хотя бы по одному модулю каждого типа устройства.

Однако построение многомашинных систем из серийно выпускаемых ЭВМ с их стандартными операционными системами значительно проще, чем построение МПС, требующих преодоления определенных трудностей, возникающих при реализации общего поля памяти, и, главное, трудоемкой разработки специальной операционной системы.

Многопроцессорные системы представляют собой основной путь построения ВС сверхвысокой производительности. При создании таких ВС возникает много сложных проблем, среди которых отметим осуществление быстродействующих экономических по аппаратурным затратам межмодульных связей, снижение потерь производительности из-за конфликтов при попытках нескольких процессоров использовать один и тот же ресурс (например, память). Указанные вопросы необходимо учитывать при выборе структуры МПС.

Многопроцессорные и многомашинные ВС, создаваемые путем комплексирования оборудования нескольких серийных ЭВМ, часто называют *вычислительными комплексами* (ВК). Термин ВК часто используют применительно к ВС, управляющей каким-либо объектом.



• На основе многопроцессорности и модульного принципа построения других устройств системы возможно создание *отказоустойчивых систем*, или, другими словами, *систем повышенной живучести*.

Многомашинные и многопроцессорные системы могут быть *однородными и неоднородными*. Однородные системы содержат однотипные ЭВМ или процессоры. Неоднородные ММС состоят из ЭВМ различного типа, а в неоднородных МПС используются различные специализированные процессоры, например процессоры для операций с плавающей точкой, для обработки десятичных чисел, процессор, реализующий функции операционной системы, процессор для матричных задач и др.

Многопроцессорные системы и ММС могут иметь *однуровневую и иерархическую (многоуровневую) структуру*. В первом случае процессоры (машины) системы образуют один общий уровень обработки данных (рис. 15.1), а во втором (рис. 15.2) система содержит отдельные машины (процессоры) для выполнения различных уровней обработки информации. Обычно менее мощная машина (*машина-спутник*) берет на себя ввод информации с различных терминалов и ее предварительную обработку, разгружая от этих сравнительно простых процедур основную, более мощную ЭВМ, чем достигается увеличение общей производительности (пропускной способности) комплекса. В качестве машин-спутников используют малые или микроЭВМ.

Важной структурной особенностью ВС является *способ организации связей между устройствами (модулями) системы*. Он непосредственно влияет на быстроту обмена информацией между модулями, а следовательно, на производительность системы, быстроту ее реакции на поступающие запросы, приспособленность к изменениям конфигурации и, наконец, размеры аппаратных затрат на осуществление межмодульных связей. В частности, от организации межмодульных связей зависят частота возникновения конфликтов при обращении процессоров к одним и тем же ресурсам (в первую очередь, к модулям памяти) и потери производительности из-за конфликтов.

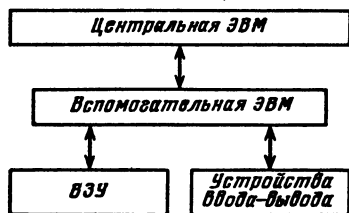


Рис. 15.2. Многомашинный вычислительный комплекс иерархической структуры

Используются следующие способы организации межмодульных (межустройственных) связей:

многоуровневые связи, соответствующие иерархии интерфейсов ЭВМ;

общая шина;

регулярные связи между модулями;

коммутатор межмодульных связей.

Принципы организации МПС и ММС существенно отличаются в зависимости от их назначения. Поэтому целесообразно различать:

1) ВС, ориентированные в первую очередь на повышение надежности и живучести;

2) ВС, ориентированные в первую очередь на достижение сверхвысокой производительности (суперЭВМ).

## **15.2. Методы и средства организации многомашинных и многопроцессорных вычислительных комплексов на основе ЭВМ общего назначения (ЕС ЭВМ)**

ЭВМ Единой системы первоначально создавались в основном как однопроцессорные универсальные по назначению машины, содержащие большой набор периферийных устройств. В этих ЭВМ используется иерархия разнообразных интерфейсов для связи между различными группами устройств (см. § 11.3), что определяет специфику построения многомашинных и многопроцессорных комплексов (ММК и МПК) в ЕС ЭВМ.

В соответствии с этой иерархией интерфейсов в ЕС ЭВМ для комплексирования машин в ММК и МПК создан набор средств комплексирования или *системных средств*, обеспечивающих возможность обмена информацией между машинами. Связь между машинами может осуществляться несколькими способами (на нескольких уровнях структуры машин):

1) на уровне процессоров — через интерфейс прямого управления (в ММК) и через общее поле оперативной памяти (в МПК);

2) на уровне каналов ввода-вывода — при помощи адаптеров канал — канал (используются в ММК);

3) на уровне внешних ЗУ — через общие поля внешних ЗУ, создаваемых с помощью разделенных устройств управления (с встроенными в них двухканальными переключателями) запоминающими устройствами на дисках и лентах (применяются в ММК и МПК).

В качестве примера на рис. 15.3 приведена упрощенная структура двухмашинного вычислительного комплекса ВК2М46,

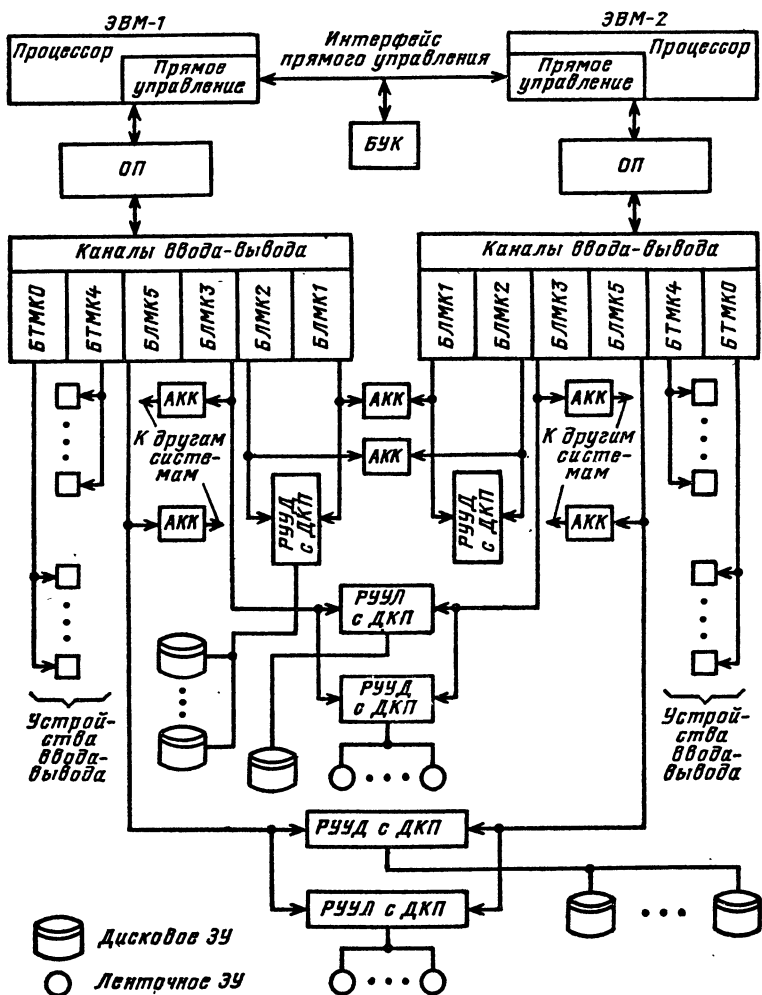


Рис. 15.3. Средства системной организации в ЕС ЭВМ. Структура двухмашинного комплекса ВК2М46:

БТМК и БЛМК — байт и блок-мультиплексные каналы; АКК — адаптер канал — канал; РУУД и РУУЛ — разделенные устройства управления соответственно дисковыми и ленточными ЗУ; ДПК — двухканальный переключатель; БУК — блок управления комплексом

состоящего из двух ЭВМ ЕС-1046, в которых используются указанные выше средства [68].

Понятие *интерфейса прямого управления* включает в себя систему шин, сигналов и *команд прямого управления*, предназначенных для выполнения процедуры обмена информацией между процессорами, при этом используется механизм внешних прерываний.

В ЕС ЭВМ имеются две команды прямого управления (формата *SI*): «Прямая запись» и «Прямое чтение». При выполнении команды «Прямая запись» процессор данной ЭВМ выставляет на выходной информационной шине прямого управления сигналы, соответствующие коду адресуемого командой байта из своей ОП, и одновременно посылает в другую ЭВМ сигнал запроса внешнего прерывания. Процессор другой ЭВМ по сигналу прерывания переходит к выполнению команды «Прямое чтение», принимает байт данных со своей входной информационной шины и помещает его в свою ОП по адресу, указанному в этой команде.

Связь через интерфейс прямого управления используют главным образом для передачи информации, управляющей и синхронизирующей работу комплекса.

*Разделенные устройства управления* (РУУ) обеспечивают доступ обеих машин к некоторым общим периферийным устройствам.

*Двухканальные переключатели* (ДКП) позволяют подключать блоки управления ВЗУ к двум каналам различных машин и создавать общие поля внешней памяти на дисках и лентах. Часть ВЗУ может оставаться в индивидуальном пользовании отдельных машин.

В изображенном на рис. 15.3 комплексе используются разделенные устройства управления со встроенными двухканальными переключателями.

*Адаптер канал — канал* (АКК) — стартстопное устройство, позволяющее производить обмен большими информационными массивами со скоростью, соответствующей пропускной способности менее быстродействующего из двух им связываемых каналов. Адаптер канал — канал при работе в монопольном режиме на расширенный двухбайтовый интерфейс может обеспечивать обмен информацией между машинами со скоростью до 3 Мбайт/с, а при работе на однобайтовый интерфейс — со скоростью до 1,5 Мбайт/с. Адаптер канал — канал подключается к каналу и управляется им как обычное УПУ, при этом каждая из связанных адаптером ЭВМ по отношению друг к другу является ПУ.

Отметим, что в ВК, представленном на рис. 15.3, в состав каждой ЭВМ входят три АКК. Один из них может работать на

одно- или двухбайтовый интерфейс, а другие — только на однобайтовый.

Повышение надежности и возможность быстрой замены отказавшей машины достигаются дублированием информации в ОП и общих ВЗУ, формированием пакета запросов и файлов входных данных в общих ВЗУ, использованием АКК для быстрого «выравнивания» содержания ОП машин при вводе отремонтированной ЭВМ в комплекс.

В индивидуальных ВЗУ хранятся операционная система машины и программы и данные фоновых (не основных) задач, выполняемых в режиме параллельной работы, когда другая ЭВМ обрабатывает основную программу.

Комплекс снабжают БУК, подключаемым к интерфейсу прямого управления. Блок управления комплексом, беря на себя функции пультов управления отдельных ЭВМ, обеспечивает задание конфигурации комплекса и режимов его работы, а также индикацию состояний отдельных ЭВМ.

Основными режимами работы двухмашинных комплексов являются следующие: дуплексный (режим нагруженного резервирования), параллельная работа машин (распределение задач между машинами), автономная работа машин (например, с выводом одной машины на профилактические испытания или ремонт), со второй машиной в горячем или холодном резерве.

В дуплексном режиме обе машины выполняют одни и те же операции над одной и той же информацией. Работа обеих машин синхронизируется электронными часами одной из машин (ведущей). В памяти обеих машин в каждый момент времени находится одна и та же информация. В определенные моменты, например после выполнения каждой команды или некоторых частей программы, результаты вычислений в обеих машинах сравниваются; при их совпадении работа машин продолжается, и ведущая машина выдает результаты во внешнюю среду, например управляющие воздействия на объект управления. Выходы другой машины (ведомой) при этом блокированы.

При обнаружении несовпадения в результатах обработки автоматически определяется неисправная ЭВМ (по сигналу системы автоматического контроля или при помощи тестов), которая выводится на ремонт, а исправная ЭВМ продолжает работу под контролем встроенной в ЭВМ системы автоматического контроля, при этом, если она ранее была ведомой, ее выходы разблокируются.

После устранения отказа отремонтированная машина вводится в дуплексный режим в качестве ведомой. Предварительно в ее память переписывается содержимое памяти ведущей машины.

Очевидно, что если средняя продолжительность ремонта (среднее время восстановления после отказа составляет около 0,5 ч) существенно меньше среднего интервала времени между возникновениями неисправностей (см. гл. 12), то в дуплексном режиме значительно повышается общая надежность комплекса.

При работе комплекса в дуплексном режиме почти на порядок увеличивается средняя наработка на отказ по сравнению со средней наработкой одной машины.

Если этого оказывается недостаточно, то в комплекс добавляется третья ЭВМ. В трехмашинном ВК возможна дуплексная работа любой пары ЭВМ, а третья служит резервной, автоматически замещающей в дуплексе вышедшую из строя или выводимую на профилактические испытания машину. При этом используются дополнительные системные средства в виде групповых коммутаторов, подключающих линии интерфейсов ввода-вывода к соответствующим каналам.

Двухмашинные комплексы используют также, когда необходимо получить производительность и пропускную способность, большие, чем может обеспечить одна ЭВМ, например, чтобы снять пиковые нагрузки по обработке или вводу информации. В таких случаях машины комплекса, обмениваясь друг с другом информацией, выполняют разные программы (задачи) или по одним и тем же программам обрабатывают разные данные (режим параллельной работы машин).

### *Проблемы построения многопроцессорных вычислительных комплексов в ЕС ЭВМ*

В многопроцессорных ВК создается общее поле ОП, обеспечивающее наибольшую скорость обмена данными между основными устройствами комплекса — процессорами, модулями памяти, каналами ввода-вывода.

При построении многопроцессорных ВК в ЕС возникает ряд довольно сложных проблем, связанных с организацией многопроцессорности на основе штатных процессоров мультитипограммных однопроцессорных ЭВМ универсального назначения, недостаточно приспособленных для этой цели. Основные затруднения связаны с осуществлением общего поля ОП.

*Разнесение зон фиксированных ячеек.* В машинах ЕС ЭВМ выделена зона фиксированных ячеек — постоянно распределенная область памяти (первые 4 Кбайт), ячейки которой имеют определенное функциональное назначение, а их адреса зафиксированы для определенных процедур, например адреса «старых» и «новых» ССП, адресного слова канала и др. В общем случае при работе процессоров в МПК их зоны фиксированных ячеек содержат неодинаковую информацию. Поэтому при по-

строении общего поля ОП возникает необходимость выделения каждому процессору своей зоны фиксированных ячеек, не пересекающейся с зонами других процессоров, и создания механизма, который обращения данного процессора по адресу из начальной (4 Кбайт) области трансформирует в соответствующий адрес выделенной процессору зоны. Этот механизм называется префиксацией.

*Механизм префиксации.* Каждому процессору МПК присваивается префикс — 12-разрядный код, задающий 12 старших разрядов начального адреса (младшие 12 разрядов начального адреса — нули) выделенных процессору 4 Кбайт в общем поле ОП в качестве его постоянно распределенной области памяти. Префикс определяет значение смещения в общем поле ОП адресов 0—4095 для данного процессора. Процессоры снабжаются программно-загружаемыми регистрами префикса. Префиксация производится блоком управления памятью непосредственно перед обращением и состоит в следующем преобразовании адреса: старшие 12 разрядов адреса заменяются префиксом, если эти разряды равны нулю, заменяются нулями, если эти разряды равны префиксу (обратная префиксация), не изменяются в других случаях. Младшие 12 разрядов адреса не подвергаются изменению.

*«Выравнивание» содержимого собственных специальных памяти процессоров.* Процессоры ЕС ЭВМ имеют собственные быстродействующие память ключей защиты (ПКЗ), ассоциативную память страничной таблицы (блок быстрой переадресации), буферную память. Если один процессор с санкции операционной системы произвел изменение ключа защиты блока (страницы) или строки в страничной таблице в своих соответствующих памяти, то аналогичные изменения должны производиться и в памяти других процессоров. В противном случае могут возникнуть серьезные нарушения в работе МПК. Отметим, что применение в устройстве управления памятью одной для всего поля ОП памяти ключей защиты оказывается нерациональным из-за недопустимого возрастания частоты конфликтов при обращении к памяти.

Нарушения в работе системы могут возникнуть, если один процессор изменил данные в ОП, а другой, выполняя программу, воспользовался устаревшей копией этих данных в своей буферной памяти. Поэтому при изменении процессором некоторых данных в ОП копии этих данных, ставшие неактуальными в буферных памяти других процессоров, должны объявляться недействительными. Аналогично, если процессор изменил строку страничной таблицы в ОП, для соответствующей физической страницы должна быть аннулирована строка в блоках быстрой переадресации всех процессоров. «Выравнивание» информации в собственных памяти процессоров достигается с помощью соответствующих управляющих сигналов.

*Межпроцессорная связь в МПК* помимо упомянутых выше команд прямого управления достигается с помощью специальной команды *сигнал процессору* (формат RS), задающей номер вызываемого процессора

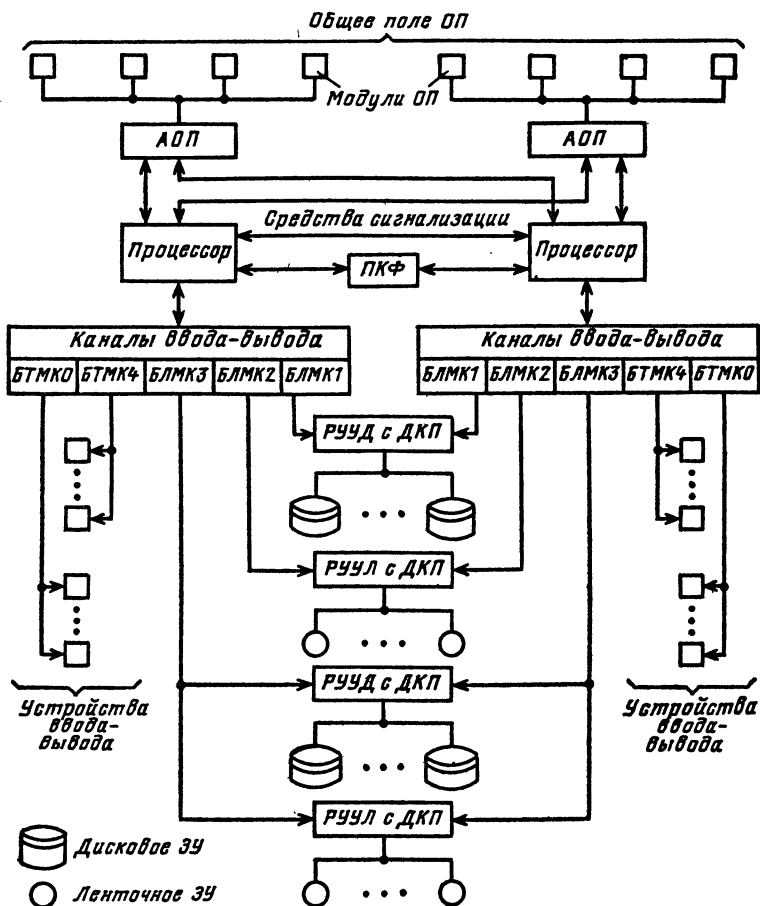


Рис. 15.4. Способы и средства системной организации в ЕС ЭВМ. Структура двухпроцессорного комплекса ВК2П46:

БТМК и БЛМК — байт- и блок-мультиплексные каналы; АОП — адаптер оперативной памяти; РУУД и РУУЛ — разделенные устройства управления соответственно дисковыми и ленточными ЗУ; ДКП — двухканальный переключатель; ПКФ — пульт конфигурации комплекса

(находится в общем регистре, указанном в поле  $R_3$  команды) и приказ (находится в полях  $B_2$ ,  $D_2$  команды). С помощью соответствующих приказов вызываемый процессор может быть установлен в состояние *стоп* или *работа*; в него может быть произведена начальная загрузка микропрограмм, могут быть вызваны экстренное внешнее прерывание и некоторые другие процедуры.

**Синхронизация часов.** В МПК должна быть единая служба астроно-



мического времени, что достигается соответствующей синхронизацией часов процессоров путем одновременного изменения значений младших разрядов всех часов.

*Общее поле памяти многопроцессорных вычислительных комплексов* в ЕС ЭВМ реализуется либо с помощью многоходовых адаптеров модулей памяти, либо с помощью коммутаторов межмодульных связей (коммутаторов перекрестных связей). Так, например, в двухпроцессорном ВК на основе ЭВМ ЕС-1046 общее поле ОП (до 8 Мбайт) строится с использованием двухходовых адаптеров основной памяти (АОП) (рис. 15.4) [68].

В ЕС ЭВМ на основе рассмотренных средств комплексирования можно создавать многомашинные комплексы, содержащие до четырех ЭВМ, причем каждая из них может быть двухпроцессорной.

Важным направлением повышения производительности ЭВМ и систем общего назначения является включение в их состав специализированных процессоров, ориентированных на определенный круг задач. Примером является «матричный процессор»<sup>1</sup>, подключаемый через двухбайтный интерфейс ввода-вывода к блок-мультиплексному каналу. Процессор воспринимает и по-своему интерпретирует команды ввода-вывода и управляющие слова канала, при этом обеспечивается возможность параллельных вычислений в процессоре ЭВМ и матричном процессоре. Матричный процессор предназначен для выполнения над потоком входных данных операций свертки, корреляции, преобразования Фурье, операций над матрицами и векторами. В матричном процессоре при выполнении арифметических операций реализуется конвейерная обработка (совмещаются операции умножения, сложения с нормализацией, выдача результатов). ЭВМ ЕС-1046 в штатном составе имеет производительность 1300 тыс. операций/с, а при подсоединении матричного процессора и решении задач с многократно повторяющимися действиями над группой данных (например, матрицами) производительность возрастает до 30 млн. эквивалентных операций/с.

### **15.3. Многопроцессорные и многомашинные комплексы с общей шиной (СМ ЭВМ)**

Организация межмодульных (и межмашинных) связей на основе общей шины является одним из распространенных способов построения многопроцессорных и многомашинных вычислительных комплексов.

---

<sup>1</sup> Здесь термин «матричный» отражает ориентацию этого специализированного процессора на решение задач матричной алгебры, а в § 15.5 этот же термин характеризует структуру многопроцессорной системы.

Общая структура многопроцессорной системы с общей шиной изображена на рис. 15.5. Входящие в состав системы процессоры, модули ОП, блоки управления периферийными устройствами (ВЗУ и устройствами ввода-вывода) подключены к одной общей шине, состоящей из линий, по которым передаются информационные и управляющие сигналы.

Одновременно через общую шину может передаваться информация только между двумя устройствами, т. е. шина используется подсоединенными к ней устройствами *в режиме разделения времени*. Это является причиной возникновения конфликтов, при которых несколько устройств претендуют на занятие шины. Наличие конфликтов на общей шине вызывает простои оборудования и уменьшает производительность системы. Частоту возникновения конфликтов и потери из-за них производительности удастся снизить, если наряду с общей доступной всем процессорам памяти иметь для каждого процессора небольшую местную (локальную) оперативную память, к которой процессор может обращаться непосредственно, минуя общую шину [26].

Из-за конфликтов в общей шине, снижающих производительность системы, организация межмодульных связей на основе общей шины не находит применения в МПС, ориентированных на достижение высокой производительности. Однако общая шина широко применяется при построении многопроцессорных и многомашинных вычислительных комплексов, предназначенных для обеспечения повышенной надежности и живучести, необходимой производительности и пропускной способности при работе в режиме реального времени в различных системах управления технологическими процессами, автоматизации экспериментов и испытаний и др. Применению шинной структуры способствует то, что в оборудовании этих комплексов широко используются малые и микроЭВМ, микропроцессорные средства, в которых реализованы различные модификации интерфейса «общая шина».

В СМ ЭВМ при построении на основе аппаратуры малых ЭВМ многопроцессорных и многомашинных комплексов с шинной структурой в качестве средств комплексирования (систем-

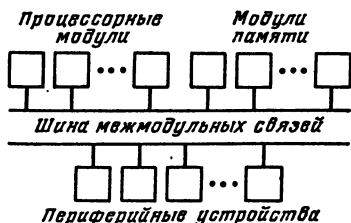


Рис. 15.5. Многопроцессорные ВС с общей шиной межмодульных связей

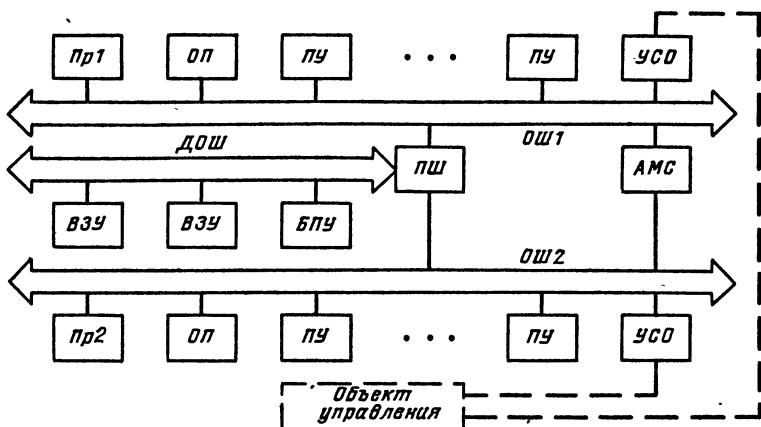


Рис. 15.6. Пример использования системных средств СМ ЭВМ — переключателя шины (ПШ) и адаптера межшинной связи (АМС) для построения многомашинных и многопроцессорных комплексов:

Пр1, Пр2 — процессоры; УСО — устройство связи с объектом управления; БПУ — быстродействующее печатающее устройство

ных средств) служат переключатель шины (ПШ) и адаптер межшинной связи (АМС) (рис. 15.6) [40, 56].

**Переключатель шины.** С помощью ПШ дополнительная общая шина ДОШ с подсоединенным к ней оборудованием может быть подключена к общей шине одного из двух процессоров комплекса или устанавливаться в отключенное состояние. Дополнительная общая шина допускает подключение любых устройств (модулей памяти, ПУ и др.), кроме процессоров.

После подключения ДОШ становится участком соответствующей общей шины комплекса и обмен с подсоединенным к ней оборудованием как программно-управляемый, так и с прямым доступом производится по обычным процедурам интерфейса общей шины.

**Адаптер межшинной (межинтерфейсной) связи** предназначен для осуществления быстрого доступа процессора и ПУ одного комплекса к памяти и ПУ другого комплекса. Здесь под комплексом понимается совокупность устройств, подключенных к одной общей шине.

Наличие АМС позволяет реализовать совместное решение комплексами задачи, иерархическую организацию обработки данных с обменом данными между главным и подчиненным комплексами, различные режимы резервирования оборудования.

Межмашинная связь с помощью АМС основана на автоматическом преобразовании адресов в командах инициировавшего

межкомплексный обмен информацией «комплекса-задатчика» в адреса «комплекса-исполнителя». Это преобразование осуществляется с помощью *окна интерфейса*, представляющего собой фиксируемую при изготовлении комплекса часть неиспользуемых им адресов. Размер окна ограничивает размер блока данных, передаваемых из одного комплекса в другой.

Адаптер позволяет производить обращение устройства одного комплекса с помощью команд процессора через окно к устройствам другого комплекса для выборки команды, записи и считывания данных, реализуемых в этом случае в другом комплексе как передача с прямым доступом и выполнение запроса программного прерывания из другого комплекса.

Процессор задает область адресов, к которым через свое окно может обратиться другой комплекс, имеет возможность запретить обращение из другого комплекса к устройствам своего комплекса или только запретить запись в свои устройства, запретить прерывание от другого комплекса.

Входящий в состав АМС регистр исполнительного адреса (младшие его разряды носят название регистра смещения) содержит сформированный окном исполнительный адрес обращения к другому комплексу. Младшие разряды этого адреса («смещение») берутся из адреса окна, выставленного на своей шине комплексом — инициатором обмена, а старшие — из регистра адреса «перемещения», устанавливаемого другим комплексом при подготовке обмена через окно.

Организация связи между комплексами через окна интерфейса поясняется на рис. 15.7. Адресное пространство 0—64 К обоих комплексов используется модулями памяти. Неиспользуемые 8 К адресов (64—72 К) предоставлены для окна интерфейса (фиксируются при изготовлении комплекса). При подготовке передачи комплекс-исполнитель засылает перемещение в АМС, устанавливая таким образом связь окна другого комплекса с областью своих адресов. Так, на рис. 15.7 первый

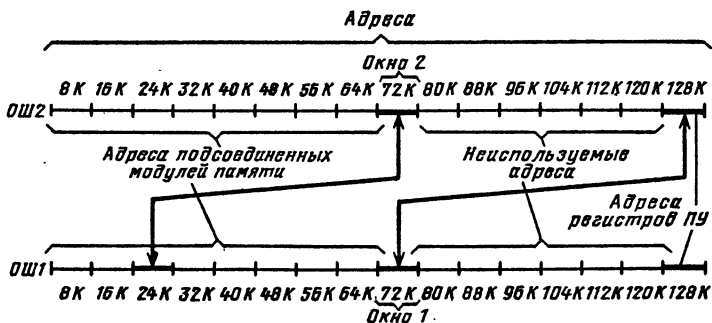


Рис. 15.7. Организация межшинной связи через окно интерфейса

комплекс через свое окно связан с ПУ второго комплекса, а второй комплекс — через свое окно с областью 24—32 К памяти первого комплекса. Время преобразования адреса в окне составляет примерно 300 нс.

Рассмотрим особенности процедуры выполнения процессором *Пр2* операции с использованием в качестве операнда слова с адресом *А* из памяти первого комплекса. Предварительно *Пр2* устанавливает в АМС перемещение — группу старших разрядов адреса, задающих область памяти с адресом *А*, и выдает команду чтения по адресу *А'*, принадлежащему окну 2. Выполняя эту команду, *Пр2* устанавливает на *ОШ2* запрос на чтение по адресу *А'*. Окно 2 с помощью перемещения, предварительно установленного *Пр2*, трансформирует этот адрес в адрес *А* области памяти первого комплекса и выставляет на *ОШ1* запрос прямого доступа. Запрашиваемая информация (слово) из адреса *А* памяти первого комплекса передается во входной буферный регистр АМС. Затем это слово из выходного буферного регистра пересылается по *ОШ2* в процессор *Пр2*.

Аналогично одной командой выполняется передача блока информации от одного комплекса другому. Например, для передачи в память первого комплекса блока данных с ЗУ на дисках второго комплекса процессор *Пр1* предварительно устанавливает перемещение для задания области своей памяти (24—32 К), куда будут записываться данные с ЗУД, и выдает на *ОШ1* запрос на чтение по соответствующему адресу из области адресов своего окна. Окно 1 преобразует этот адрес в адрес регистра ЗУД и выставляет на *ОШ2* запрос прямого доступа для передачи блока данных с ЗУД по адресам окна 2, при этом адреса окна 2 преобразуются в адреса области (24—32 К) памяти первого комплекса, установленной перемещением.

На одной ОШ может быть организовано несколько окон (выделено несколько областей неиспользуемых адресов) для связи с несколькими комплексами.

На рис. 15.6 показан пример использования *ПШ* и *АМС* для построения на основе оборудования СМ ЭВМ многомашинных управляющих вычислительных комплексов. В представленной структуре реализуются общее поле памяти и общее поле ВЗУ, двухпроцессорная параллельная обработка данных и резервирование наиболее важных для управления объектом устройств — процессора, памяти, УСО и некоторых ПУ.

#### **15.4. Особенности организации отказоустойчивых многопроцессорных вычислительных комплексов**

Перспективы широкого применения построенных на основе микроЭВМ и микропроцессоров управляющих вычислительных устройств (УВУ) и комплексов (УВК) для управления в реальном масштабе времени различными технологическими процесса-

ми и установками выдвигают на первый план проблему обеспечения надежности их работы. Управляющие комплексы и устройства, работающие в составе различных АСУ ТП, должны обеспечивать длительное достоверное (безошибочное) функционирование без остановки технологического процесса при сбоях и отказах в оборудовании УВУ и УВК. В ряде случаев предъявляются жесткие требования в отношении недопустимости перерывов или задержек в осуществлении УВУ и УВК управляющих функций.

Указанную проблему призваны решать *отказоустойчивые УВУ и УВК*, т. е. устройства и комплексы, которые сохраняют работоспособность при выходе из строя отдельных элементов, узлов или модулей.

Вопросам обеспечения отказоустойчивости уделялось довольно много внимания в рамках ЭВМ III поколения, и при этом получены определенные результаты; например, в составе ЕС ЭВМ и СМ ЭВМ созданы многомашинные и многопроцессорные комплексы (см. § 15.2 и 15.3), хотя ЭВМ ЕС, как, впрочем, и машины СМ ЭВМ, сначала проектировались в основном для одномашинного и однопроцессорного режимов работы. Новые возможности, связанные с небольшими размерами и сравнительно низкой стоимостью микропроцессорных и других БИС, позволяют существенно продвинуться в этом направлении путем создания ЭВМ и комплексов со *специальной архитектурой, ориентированной на достижение отказоустойчивости*.

Основным принципом, на основе которого создаются отказоустойчивые ЭВМ, системы и комплексы, является аппаратурная избыточность. В настоящее время определилось несколько направлений в организации отказоустойчивых вычислительных комплексов. При сравнении этих направлений следует учитывать размеры аппаратурной избыточности (хотя в ряде случаев это может и не иметь определяющего значения), меру обеспечения достоверности результатов обработки комплексом данных, степень исключения влияния сбоев и отказов в аппаратуре на изменение временной диаграммы выдачи результатов обработки данных на объект управления.

Надежность УВУ и УВК при необходимости длительного функционирования весьма сильно зависит от степени их приспособленности к техническому обслуживанию (в первую очередь от ремонтпригодности) в конкретных условиях эксплуатации. Этот показатель может в ряде случаев оказаться основным при выборе варианта построения отказоустойчивых УВУ и УВК.

Организация отказоустойчивого УВК (его структура, динамическое изменение конфигурации и т. п.) должна быть «прозрачной» для пользователя в том смысле, что пользователь при

взаимодействии с УВК и при подготовке и отладке программ для него должен воспринимать комплекс как обычную однопроцессорную ЭВМ с неизменной конфигурацией.

*Отказоустойчивые УВК с автоматической реконфигурацией.* Обычно функции (программы), реализуемые УВК, неравнозначны, и временная утрата некоторых второстепенных функций при сохранении основных жизненно важных допустима. Поэтому для таких УВК основной характеристикой надежности следует считать не среднее время наработки на отказ, используемое для оценки надежности ЭВМ общего назначения, а вероятность сохранения на заданном интервале любой из основных функций вследствие отказов отдельных модулей системы. Эта вероятность может быть принята за меру живучести комплекса. Особенности построения отказоустойчивых УВК с автоматической реконфигурацией описаны в [51].

Отказоустойчивость достигается введением избыточного оборудования и логической организацией, обеспечивающей при отказах в оборудовании *автоматическую реконфигурацию* системы для сохранения жизненно важных функций, возможно, ценой утраты второстепенных. Системы, обладающие указанной способностью, иногда называют «системами с элегантной деградацией».

Основными принципами построения отказоустойчивых комплексов на основе автоматической реконфигурации являются *многоустройство* (в том числе многопроцессорность), *общие поля процессоров, оперативной памяти, каналов и периферийных устройств, динамическое распределение функций между однотипными устройствами.*

*Многоустройство* предполагает, что система должна содержать несколько экземпляров однотипных устройств (процессоров, модулей ОП, каналов и др.), при этом должна быть обеспечена избыточность устройств всех типов по отношению к минимальному набору, необходимому для выполнения жизненно важных функций.

*Динамическое распределение функций* означает, что программы не привязаны жестко к процессорам, каналам и ПУ. Заранее неизвестно, какое из однотипных устройств будет выполнять данную функцию. Более того, работа может быть начата на одном, продолжена на другом и закончена на третьем устройстве.

Наличие общих полей устройств и динамического распределения функций позволяет комплексу сохранять работоспособность, пока имеется хотя бы по одному исправному устройству каждого типа.

Практическое осуществление отказоустойчивых УВК с авто-

матической реконструкцией требует реализации рассматриваемых ниже логических свойств.

*Многосвязные интерфейсы* позволяют многовходовым устройствам связываться друг с другом по независимым шинам.

*Динамическое распределение (диспетчирование) программ.* В ОП организуются очереди для программ (работ), имеющих различные уровни приоритета. Аппаратура выделяет среди процессоров кандидата на прерывание, которым является процессор, обрабатывающий программу наименьшего приоритета.

Аппаратура непрерывно сравнивает приоритет программы процессора — кандидата на прерывание с приоритетом программ, находящихся в очередях. Если в очереди появляется программа большего приоритета, процессор-кандидат прерывает свою программу, заносит ее в очередь соответствующего приоритета и начинает выполнять программу из непустой очереди наибольшего приоритета. Обработка прерванной программы будет продолжена, причем необязательно на том же процессоре, когда ее приоритет станет выше приоритета программы — кандидата на прерывание. Следует обратить внимание на то, что здесь образуется общая очередь к освобождающимся процессорам.

Описанный механизм обеспечивает в каждый момент времени выполнение наиболее приоритетных программ на наличных исправных процессорах.

*Системы автоматического контроля правильности работы устройства и диагностирования* — см. гл. 12.

*Иерархия рестартов (восстановлений)* программ при сбоях и отказах. Для уменьшения потерь времени на восстановление программ после сбоев и отказов следует иметь возможность восстанавливать работоспособность программ на разных уровнях: 1) сбой и отказы в ОП; б) сбой в процессорах и каналах; в) отказы в процессорах и каналах; г) сбой и отказы в периферийных устройствах.

В ОП сбой и отказы исправляются с помощью самокорректирующего кода Хэмминга (см. гл. 12).

*Микрокомандный рестарт.* Средства контроля проверяют правильность каждой микрооперации, и при неправильном выполнении микрокоманда повторяется. Для микрокомандного рестарта необходимо, чтобы в процессе исполнения микрооперации сохранялись ее операнды. Это требует дополнительного оборудования (регистров и др.) и приводит к некоторому уменьшению скорости работы процессора.

Если при повторении микрокоманда выполнится правильно, обработка программ на устройстве продолжается. Если повторение микрокоманды установленное число раз не дает правильного результата, фиксируется отказ.



*Рестарт при отказе процессора* (рис. 15.8). После фиксации системой контроля отказа состояния регистров неисправного процессора запоминаются в выделенной для этого области памяти. Неисправный процессор посылает сигнал отказа во все другие процессоры. Этот сигнал воспринимает процессор — кандидат на прерывание. Он заносит свою программу в соответствующую очередь программ, а из области аварийной регистрации считывает информацию о состоянии регистров отказавшего процессора и продолжает обработку программы последнего, начиная с неправильно выполненной им микрокоманды.

*Рестарт при сбоях и отказах в периферийных устройствах* состоит в повторении, начиная с ближайшей контрольной точки, подпрограммы операции ввода-вывода (канальной программы).

*Средства повышения ремонтоспособности.* Обеспечение свойства живучести системы на протяжении длительного интервала времени возможно лишь при высокой ремонтоспособности оборудования, когда обеспечиваются быстрый ремонт и ввод исправленных устройств в эксплуатацию. Повышение ремонтоспособности достигается аппаратно-программными средствами, обеспечивающими возможность без перерыва в работе исправной части оборудования УВК производить *программно-управляемое отсечение оборудования для ремонта, программно-управляемое присоединение* к комплексу вводимых в эксплуатацию устройств, *программно-управляемое изменение номиналов питающих напряжений и параметров основных сигналов* для автоматизации профилактических испытаний оборудования.

В вычислительных комплексах с автоматической реконфигурацией должна обеспечиваться независимость операционной системы и целевого программного обеспечения («прозрачность») относительно конфигурации аппаратных средств.

Достоинством отказоустойчивых комплексов с автоматической реконфигурацией является сравнительно небольшой объем избыточного оборудования. Однако в подобных системах достоверность функционирования проверяется только схемами контроля, что может оказаться недостаточным, так как схемы контроля охватывают обычно не все оборудование комплекса, во многих случаях реагируют толь-

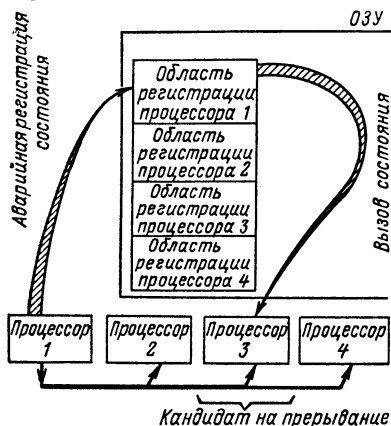


Рис. 15.8. Рестарт при отказе процессора

ко на одиночную ошибку и, кроме того, в них самих могут возникать отказы. Комплексы с реконфигурацией не позволяют полностью исключить задержки в реализации управляемых функций, так как требуется определенное время на процедуры селекции сбоев и отказов и процедуры рестартов. Приспособленность к техническому обслуживанию (ремонтпригодность и др.) фактически остается аналогичной приспособленности обычных машин.

*Вычислительные комплексы с мажоритарным управлением (тройное дублирование).* Принцип мажоритарного управления состоит в выполнении одних и тех же вычислений несколькими (более двух) ЭВМ или устройствами, сопоставлении получаемых машинами результатов по методу «голосования» и выдаче на объект управления результатов обработки, оказавшихся одинаковыми у большинства машин (устройств) комплекса. Комплекс снабжается мажоритарным устройством, в определенные моменты сравнивающим выходные данные машин и формирующим путем «голосования» общий выходной управляющий сигнал. Выход из строя одного из элементов практически не сказывается на выходных сигналах УВК. Комплексы с мажоритарным управлением в состоянии практически исключить ошибки в управлении.

Примером УВК с тройным дублированием и мажоритарным управлением может служить система «Август» (рис. 15.9) [61].

Три одинаковые управляющие микроЭВМ выполняют независимо все операции. Имеется возможность обращения для считывания одной микроЭВМ к памяти двух других. Это используется для обнаружения и исправления ошибок.

В комплексе реализуются три ступени мажоритарного выбора (рис. 15.10).

1. Каждая микроЭВМ принимает от датчиков входные сигналы через свои входные цепи. Затем микроЭВМ считывает значения входных величин, полученные другими машинами, и при

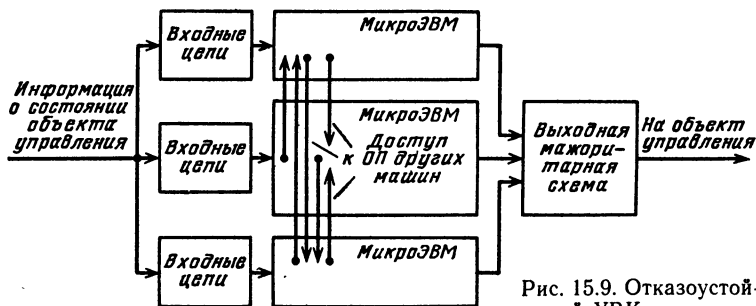


Рис. 15.9. Отказоустойчивый УВК с мажоритарным управлением

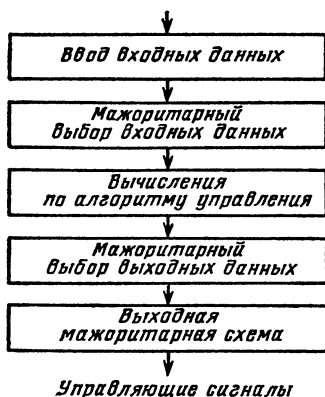


Рис. 15.10. Многоступенчатый мажоритарный выбор

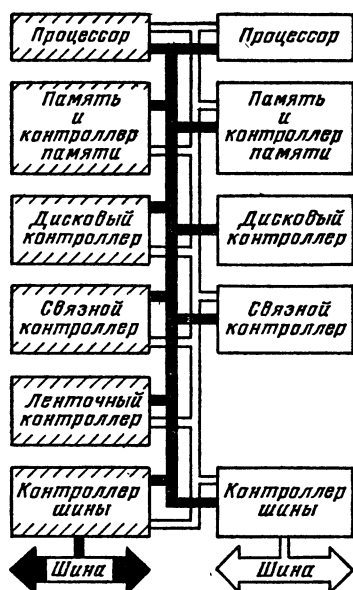


Рис. 15.11. Отказоустойчивый УВК с двухуровневым дублированием

необходимости корректирует свои входные данные по правилу «2 из 3». Таким образом, исключаются влияния неисправностей входных схем.

2. Каждая микроЭВМ (процессор) по выравненным входным данным производит одни и те же вычисления, определяя свой вариант выходных управляющих сигналов или величин для объекта управления. Затем микроЭВМ сравнивают полученные выходные результаты и выравнивают их по правилу «2 из 3».

3. Одинаковые выходные результаты выдаются через собственные интерфейсы каждой микроЭВМ на мажоритарную схему (аппаратурный мажоритарный выбор).

Достоинствами рассматриваемого подхода являются последовательный контроль и коррекция ошибок от сбоев и отказов в подсистемах ввода, обработки и вывода данных. К недостаткам помимо большего, чем при автоматической реконфигурации, объема аппаратуры можно отнести следующее: выравнивание информации и определение неисправного модуля производятся программным путем с соответствующей потерей времени и задержками в управлении. Обнаруженный неисправный модуль является достаточно сложным устройством, и дальнейшее уточнение места неисправности требует квалифицированного обслуживания.

*Отказоустойчивые комплексы с двухуровневым дублирова-*

нием (*четырёхкратное дублирование*). Небольшие размеры и низкая стоимость микропроцессорных схем позволяют реализовать многопроцессорные комплексы с обеспечением отказоустойчивости на уровне аппаратуры и полностью исключить перерывы или задержки в осуществлении комплексом управляющих функций, которые могут возникнуть при рестартах после отказов или сбоев, выполняемых программными или микропроцессорными средствами.

Этого можно достигнуть, используя двухуровневое дублирование аппаратуры, подобно тому, как это сделано в системе «Стратус» [63].

Комплекс содержит двойной комплект всех плат (модулей), попарно выполняющих одновременно одни и те же действия (первый уровень дублирования). На каждой плате размещены полностью дублированные тракты преобразования информации и схемы, контролирующие совпадение выходных сигналов в случае несовпадения (второй уровень дублирования). Структура комплекса представлена на рис. 15.11.

При обнаружении в какой-либо плате несовпадения выходных сигналов ее трактов из-за сбоя или отказа данная плата логически отключается от комплекса, а в операционную систему и на панель сигнализации оператора поступает соответствующий сигнал, при этом нормальную работу продолжает дублирующая плата и не требуется никакой специальной процедуры рестарта.

Рассматриваемый подход требует большего объема аппаратуры, чем тройное дублирование с мажоритарным управлением. Однако значительно упрощаются сравнивающие процедуры и схемы, упрощаются схемы аппаратурного контроля правильности функционирования, полностью исключаются оперативные действия программных и микропрограммных средств после сбоев и отказов (так как отказоустойчивость обеспечивается на аппаратурном уровне, отказы и сбои обнаруживаются, и их последствия блокируются непосредственно в момент их возникновения), исключаются возвраты к контрольным точкам, которые имеются при использовании программных и микропрограммных средств в процедурах рестартов. В комплексах с двухуровневым дублированием в большой степени упрощается техническое обслуживание аппаратуры, так как автоматически обнаруживаются неисправные платы и имеется возможность замены последних без остановки УВК.

При создании УВК с мажоритарным управлением или двухуровневым дублированием должны быть решены вопросы синхронизации работы дублированных плат и логических трактов.

## 15.5. Типы структур многопроцессорных ВС, ориентированных на достижение сверхвысокой производительности

Многопроцессорные системы, ориентированные на достижение сверхбольших скоростей работы (сотни миллионов или миллиарды операций в секунду), содержат десятки или сотни сравнительно простых (элементарных) процессоров с упрощенными блоками управления. Отказ от универсальности применения таких ВС и специализация их на определенном, правда, достаточно широком круге задач, допускающих эффективное распараллеливание вычислений, позволяют строить их с регулярной структурой связей между процессорами.

Удачной, на наш взгляд, является классификация Флина [77], облегчающая понимание особенностей структур и функционирования МПС, в первую очередь систем с регулярными связями. Классификация строится по признаку одинарности или множественности потоков команд и данных.

*Однопроцессорная ЭВМ.* Структура обыкновенной однопроцессорной ЭВМ (рис. 15.12, а) содержит *одинарный поток команд и одинарный поток данных (структура ОКОД)*.

*Матричная МПС — структура типа ОКМД* (рис. 15.12, б). Система содержит некоторое число одинаковых сравнительно простых быстродействующих процессоров, соединенных друг с другом и с памятью данных регулярным образом так, что образуется сетка (матрица), в узлах которой размещаются процессоры.

В системе имеются *несколько потоков данных и один общий поток команд*, другими словами, все процессоры выполняют одновременно одну и ту же команду (допускается пропуск выполнения команды в отдельных процессорах), но над разными операндами, доставляемыми процессорам из памяти несколькими потоками данных. Подобные МПС часто называют *системами с общим потоком команд*.

В МПС с общим потоком команд возникает сложная задача распараллеливания алгоритмов решаемых задач для обеспечения загрузки процессоров. В ряде случаев эти вопросы лучше решаются в конвейерной системе.

*Конвейерная МПС — структура типа МКОД* (рис. 15.12, в). Система имеет регулярную структуру в виде цепочки последовательно соединенных процессоров или специальных вычислительных блоков (СВБ), так что информация на выходе одного процессора (СВБ) является входной информацией для следующего в конвейерной цепочке. Процессоры (СВБ) образуют *конвейер (трубопровод)*. На вход конвейера *одинарный поток*

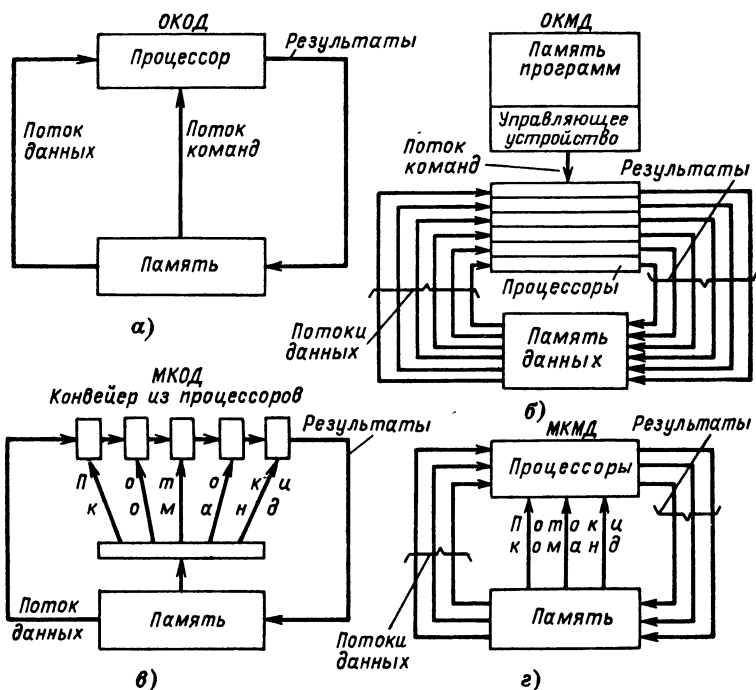


Рис. 15.12. К классификации многопроцессорных ВС по одинарности и множественности потоков команд и данных:

а — однопроцессорная ЭВМ (одиночные потоки команд и данных); б — ВС с общим потоком команд (одиночный поток команд и множественный поток данных); в — конвейерная ВС (множественный поток команд и одиночный поток команд); г — общий случай (множественные потоки команд и данных)

данных доставляет операнды из памяти. Каждый процессор (СВБ) обрабатывает соответствующую часть задачи, или операции, передавая результаты соседнему процессору (СВБ), который использует их в качестве исходных данных. Таким образом, решение задач для некоторых исходных данных разворачивается последовательно в конвейерной цепочке. Это обеспечивается подведением к каждому процессору (СВБ) своего потока команд, т. е. имеется *множественный поток команд*.

Если «трубопровод наполнен», выходной процессор (СВБ) выдает результаты для последовательности входных данных через очень короткие интервалы времени, хотя действительное время прохождения задачи через конвейер может быть существенно большим.

Конвейерная обработка описана в гл. 9, где говорилось о конвейере команд, образованном блоками процессора, реали-

зующими отдельные этапы рабочего цикла выполнения команды, и об арифметическом конвейере («конвейерном АЛУ»), состоящем из блоков, выполняющих отдельные этапы арифметических и логических операций. Эти конвейеры функционируют под управлением множественных потоков управляющих сигналов микроопераций. Внутрипроцессорная конвейерная обработка в настоящее время широко используется. Примером являются входящие в состав ЭВМ ЕС специализированные конвейерные процессоры для матричных и векторных операций (по терминологии ЕС ЭВМ называемые «матричными процессорами»). Конвейер из сложных модулей — процессоров, управляемых каждый своим потоком команд, иногда называют «макроконвейером».

*Общий случай МПС — структура типа МКМД. На рис. 14.12, г представлен общий случай структуры МПС, в которой несколько потоков данных и несколько потоков команд.*

В настоящее время конкурируют между собой главным образом две структуры построения суперЭВМ: а) с общим потоком команд; б) конвейерная (со специальными средствами обработки векторов).

### **15.6. Многопроцессорная минисуперЭВМ с общим потоком команд (ПС-2000)**

К МПС с общим потоком команд относится разработанная под руководством академика АН Грузинской ССР И. В. Прангишвили высокопроизводительная система ПС-2000, структура которой представлена на рис. 15.13 [46].

Параллельный процессор *ППр* системы содержит до 64 процессорных элементов (ПЭ) и имеет суммарную производительность до 200 млн. коротких операций. В состав системы входят также подсистемы внешней памяти *ПСВП* и мониторинговая подсистема *МтС* на основе малой ЭВМ *СМ-2М*. Общее управление всей системой производит *МтС*, которая загружает в *УУ* параллельного процессора микропрограммы, программы и константы, управляет обменом информацией между *ППр* и *ПСВП*, осуществляет связь с устройствами ввода-вывода.

Процессорные элементы выполняют операции над 24-разрядными словами. В состав каждого ПЭ входят АЛУ, память емкостью 4 или 16 Кслов с циклом обращения соответственно 640 и 960 нс, сверхоперативная память на 16 слов, локальное *УУ*. Из-за сравнительно короткого слова ПС-2000 следует отнести к минисуперЭВМ.

Соседние ПЭ соединены друг с другом регулярными связями, допускающими параллельную передачу слов. Кроме того, имеет-

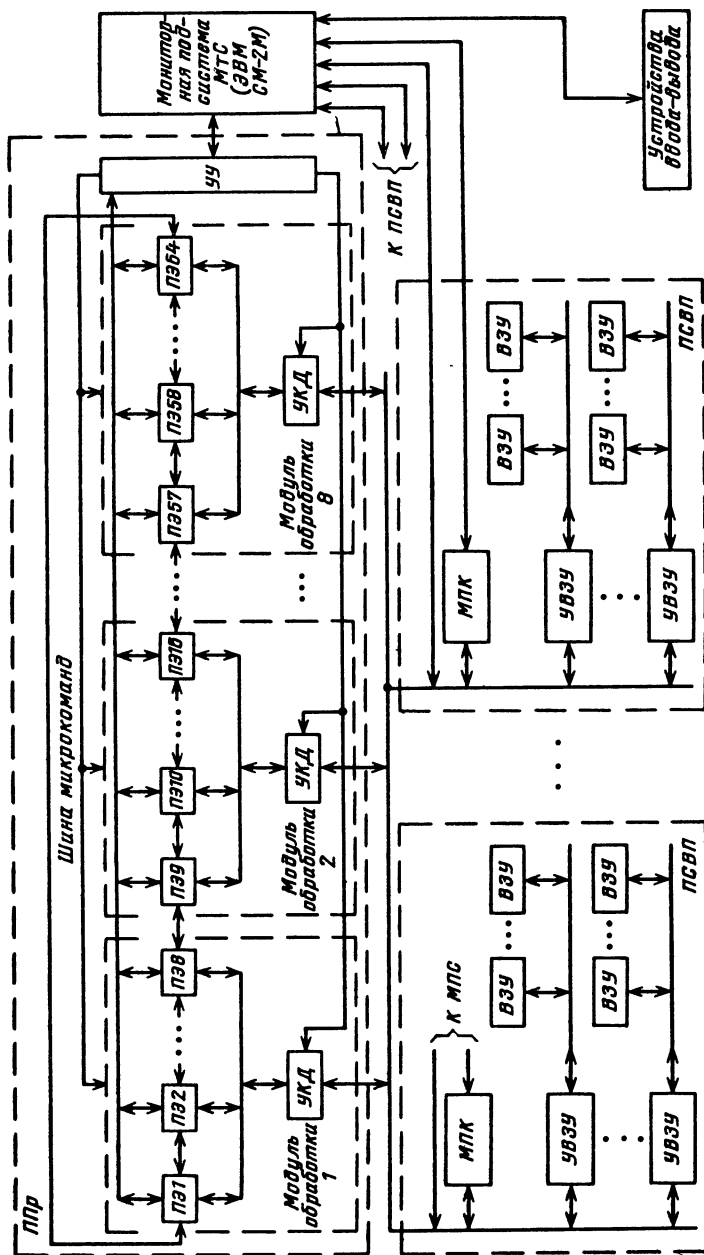


Рис. 15.13. Структура МПС с общим потоком команд ПС-2000:  
УКД — управление каналами данных; МПК — микропрограммируемый контроллер внешней памяти



ся шина («магистральный канал») для последовательной передачи слов, связывающая все ПЭ и УУ. По этой шине слово может передавать одновременно только один ПЭ или УУ, а принимать сразу все адресуемые ПЭ.

Как видно из структурной схемы, система ПС-2000 построена на основе иерархической децентрализации и распределения обработки, управления, хранения и обмена данными между ее отдельными модулями.

Эффективность использования матричных (параллельных) ВС зависит от возможностей распараллеливания программ («параллельное программирование»), что во многих случаях связано с большими трудностями.

### 15.7. Многопроцессорная суперЭВМ с коммутатором межмодульных связей («Эльбрус»)

Одним из способов построения МВС универсального назначения является использование быстродействующего коммутатора межмодульных связей (КМС). Коммутатор, приобретающий в таких системах характер центрального устройства, обеспечивает возможность установления связи любого процессора с любым модулем памяти и любым модулем управления вводом-выводом, а также любого модуля управления вводом-выводом с любым модулем памяти (рис. 15.14).

Коммутатор образует межмодульные связи путем установления межсоединений в соответствующих точках пересечения прямоугольной сетки шин. Соединения между модулями сохраняются на все время передачи данных. Возможны одновременная связь и передача данных через коммутатор между различными парами модулей системы.

Таким образом, КМС реализует пространственное разделение при коммутации соединений в отличие от временного разделения при использовании для межмодульных связей общей шины.

При наличии коммутатора существенно уменьшается число конфликтов в системе, так как сам коммутатор не порождает

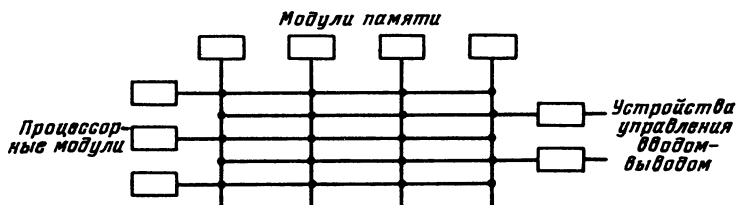


Рис. 15.14. Коммутатор межмодульных связей

конфликтов, и последние могут возникнуть только при запросах от нескольких устройств на связь с одним и тем же модулем. Однако следует иметь в виду, что коммутатор представляет собой достаточно сложное устройство, обусловленное высокими требованиями к его быстродействию, сложностью реализуемых им логических функций и необходимостью обеспечить возможность изменения за счет модульного построения самого коммутатора числа подключаемых к нему устройств (модулей) системы. Чтобы достигнуть высокой общей производительности системы, время переключения КМС должно быть существенно меньше времени выполнения операции в процессоре.

К МПС, использующим коммутатор межмодульной связи, относится вычислительная система «Эльбрус», структура которой представлена на рис. 15.15 [6, 16]. Эта универсальная по назначению система с самого начала проектировалась как многопроцессорная, причем многопроцессорность рассматривалась как средство достижения высокой производительности (до 100 млн. операций/с в «Эльбрус-2») при решении широкого круга задач, а также как средство обеспечения повышенной надежности (живучести) за счет автоматической реконфигура-

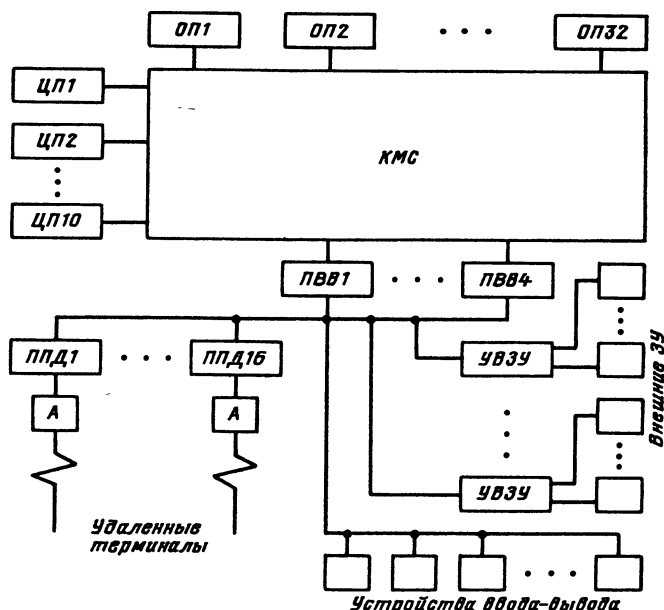


Рис. 15.15. Структура многопроцессорной ВС «Эльбрус»

ции, чему способствуют модульное построение системы и динамическое распределение ресурсов.

Рассматриваемая ВС относится к системам типа МКМД.

Система «Эльбрус» в зависимости от конфигурации содержит от 1 до 10 модулей центральных процессоров и от 4 до 32 модулей ОП (соответственно емкость памяти от 576 до 4608 Кбайт). К коммутатору межмодульных связей помимо процессоров и модулей памяти может быть подключено до четырех процессоров ввода-вывода (ПВВ).

С помощью коммутатора реализуется общее поле оперативной памяти и общее поле процессоров ввода-вывода. Процессоры ввода-вывода через дополнительный коммутатор (на схеме не показан) и соответствующие блоки управления имеют доступ к любому периферийному устройству, т. е. обеспечивается общее поле периферийных устройств.

Работа системы через линии связи с удаленными терминалами осуществляется с помощью специализированных процессоров передачи данных (ППД1—ППД16), подключенных к ПВВ1—ПВВ4, и адаптеров А.

Процессоры ввода-вывода полностью освобождают центральные процессоры от управления операциями ввода-вывода.

Общая операционная система *динамически* распределяет ресурсы между решаемыми задачами и обеспечивает параллельную и независимую работу модулей системы.

Модули системы снабжены средствами аппаратурного контроля правильности функционирования. При возникновении ошибки при передаче или преобразовании информации схемы контроля соответствующего модуля посылают сигнал, по которому операционная система производит реконфигурацию МПС, отключает неисправный модуль и выводит его в ремонт. По завершении последнего операционная система включает его в работу в составе МПС. Повышение надежности хранения информации достигается ее дублированием в устройствах памяти разного уровня.

Одной из важнейших особенностей МПС «Эльбрус» является приближение уровня машинного языка (системы команд) к уровню алгоритмического языка высокого уровня, в результате чего значительно повышается скорость трансляции программ, а следовательно, и производительность системы, облегчается труд программиста. К особенностям системы также относится широкое использование стеков, в том числе для реализации языка безадресных команд, соответствующего польской инверсной записи (см. § 9.6), повышения производительности системы, динамического распределения ресурсов (в первую очередь памя-

ти и быстрых регистров), организации обращений к процедурам, обработки прерываний.

В МПС «Эльбрус» каждое слово памяти снабжается ярлыком (тегом), представляющим собой группу дополнительных разрядов, указывающих тип данных (операнд — число с плавающей точкой, целое число, метка процедуры, дескриптор и т. п.) и формат данных (32, 64 или 128 разрядов), а также режим защиты слова. Используются дескрипторы, являющиеся гибким средством описаний хранящихся в памяти массивов информации. Дескриптор задает адрес и длину массива, тип памяти, хранящей массив, указывает, содержит ли массив данные или команды и некоторые другие характеристики массива. Обращение к информации, находящейся в памяти, производится через дескрипторы (см. § 9.6).

Теги и дескрипторы обеспечивают такие формы контекстной защиты данных, как обнаружение несоответствия кода операции и типа данных, попытки выхода за пределы массива информации, установленные дескриптором.

Вычислительный процесс организуется на основе использования стеков. В МПС «Эльбрус» реализуется *многопроцессная обработка данных*, при этом под *процессом* понимается совокупность программы задачи (или комплекса программ), используемых ею данных и выделенных ей ресурсов системы. Каждому процессу в системе соответствует свой стек в ОП. Любой процессор работает на любом стеке. Самостоятельные стеки могут представляться не только каждой задаче, но и отдельным процедурам. Таким образом, достигается широкое параллеливание решения задач.

Группа связанных между собой процессов представляется деревом стеков («кактус-стеком»).

Операционная система динамически распределяет ресурсы системы между процессами с учетом их приоритетных соотношений. Процесс может быть *активным*, если он обрабатывается системой, или *пассивным*, если он ожидает некоторого события: освобождения центрального процессора (заняты все процессоры), окончания необходимой для дальнейшей обработки процесса операции ввода-вывода, освобождения другим процессом общих данных. Пассивный процесс может находиться в *неготовом* и *готовом* для выполнения состояниях. Из очереди готовых процессов выбирается наиболее приоритетный и назначается на первый освободившийся процессор, при этом процесс получает в свое распоряжение аппаратуру процессора, в том числе указатель стека, сверхоперативную память, базовые регистры. В следующий раз при переходе из пассивного в активное состояние процесс может быть назначен на другой процессор. Допускается

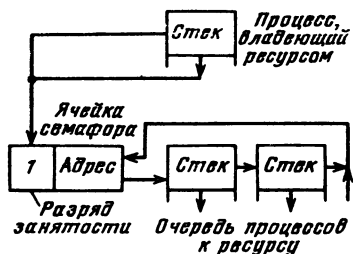


Рис. 15.16. Синхронизация процессов при помощи семафора

одновременное функционирование нескольких активных процессов, использующих общие данные и другие ресурсы, что достигается *механизмом синхронизации процессов*.

Синхронизация вычислительных процессов осуществляется с помощью *семафоров* (рис. 15.16).

Отдельным ресурсам системы — наборам данных, процессорам ввода-вывода и другим — ставятся в соответствие специальные ячейки в ОП — семафоры. Ресурс доступен процессу, если семафор открыт (в соответствующем разряде ячейки семафора 0). Занимая ресурс, процесс специальной командой «Закрыть семафор» устанавливает 1 в этот разряд ячейки.

Если семафор нужного процессу ресурса оказывается закрытым (в ячейке семафора стоит 1), процесс записывает в адресное поле семафора адрес своего стека (возможно образование у семафора очереди стеков), переходит в пассивное состояние, освобождая ЦП, который предоставляется наиболее приоритетному из очереди готовых процессов.

При освобождении ресурса, например после выполнения ПВВ запрошенного процессором ввода данных или завершения обращения другого процесса к общим данным, освобождающий ресурс процесс командой «Открыть семафор» заносит 0 в ячейку семафора соответствующего ресурса и ставит в очередь готовых процесс, адрес стека которого находится в ячейке семафора.

## 15.8. Конвейерно-векторные суперЭВМ

При решении на ЭВМ научно-технических и других задач часто встречается необходимость определения значений одной и той же функции для группы данных, расположенных в ячейках памяти (регистрах) с упорядоченными адресами (номерами). Такие наборы данных называются *векторами*.

В целях повышения производительности ЭВМ в состав ее процессора включают средства *векторной обработки данных* или в состав ЭВМ вводят специализированный *векторный сопроцессор*. Векторные средства включают векторные команды, век-

торные регистры и другую аппаратуру для реализации этих команд. *Векторные команды* позволяют одной командой предписать выполнение некоторой операции (*векторной операции*) над элементами вектора (векторов), например поэлементное сложение векторов. Таким образом, векторная команда наряду с обеспечением содержательной операции над элементами вектора берет на себя и управление вычислительным циклом.

В машине, имеющей векторные команды, выполняются также обычные команды над одиночными данными (*скалярные команды*).

Наличие векторных команд способствует повышению быстродействия процессора за счет уменьшения потерь времени на организацию вычислительного цикла.

Обычно в целях достижения повышенного быстродействия выполнение самих векторных операций конвейеризуется, причем может иметься несколько арифметических конвейеров (линий), а отдельные устройства (позиции) конвейерной линии могут, в свою очередь, содержать конвейеры для выполнения возложенных на них подфункций.

Векторные команды (обычно их сравнительно немного) имеют ряд особенностей. Их код операции не только задает подлежащую выполнению операцию, но и определяет нужную конфигурацию активных частей конвейера для данной операции. Команда указывает начало вектора (векторов) в памяти (или блоке регистров), длину вектора, размер и тип элементов вектора (целое число, число с плавающей точкой и т. д.), определяет местоположение вектора-результата.

Примерами конвейерно-векторных процессоров могут служить специализированные на выполнение векторных и матричных операций сопроцессоры<sup>1</sup>, подключаемые к машинам ЕС ЭВМ (см. гл. 13), при этом производительность машины при выполнении векторных и матричных операций увеличивается до 30 Мфлоп/с.

Конвейерно-векторные ВС в настоящее время являются одним из основных направлений при создании суперЭВМ для широкого круга задач. К ВС такого типа относятся самые быстродействующие суперЭВМ семейства CRAY (США).

На рис. 15.17 представлена упрощенная структура конвейерно-векторного процессора суперЭВМ CRAY-1. Поскольку программы решения научно-технических задач, как правило, требуют выполнения как векторных, так и скалярных операций, про-

---

<sup>1</sup> В ЕС ЭВМ эти специализированные сопроцессоры называют «матричными» — по типу решаемых задач.

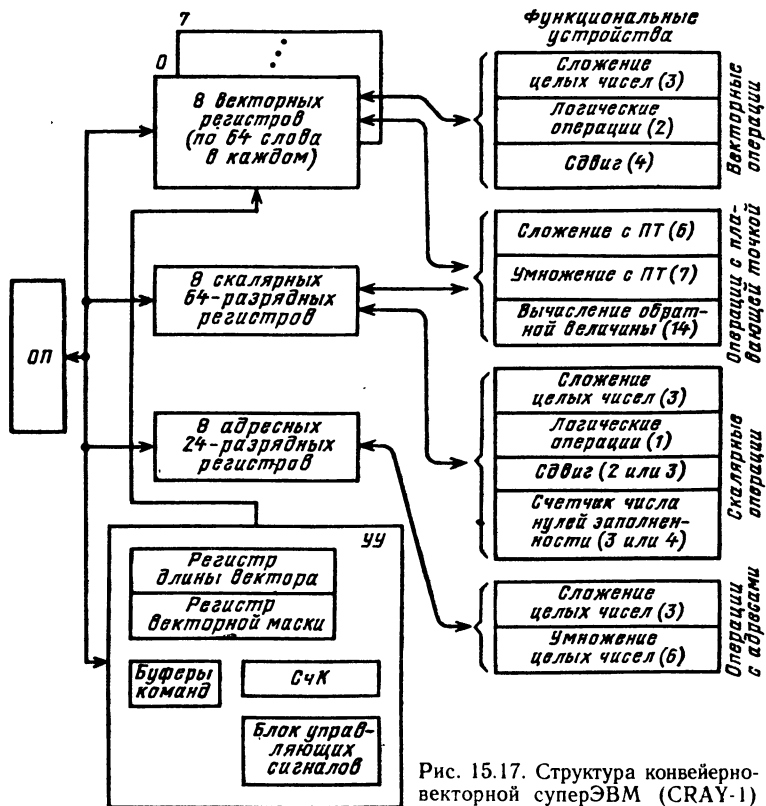


Рис. 15.17. Структура конвейерно-векторной суперЭВМ (Cray-1)

цессор имеет командные и аппаратные средства для операций обоих типов. При этом приняты специальные меры по уменьшению времени выполнения скалярных операций, чтобы последние существенно не снижали общую производительность системы. В системе реализованы конвейер команд и многочисленные конвейеры для различных арифметических и логических операций при скалярной и векторной обработках 64-разрядных слов с плавающей точкой (порядок — 15, мантисса — 49 разрядов), 64- или 24-разрядных слов с фиксированной точкой, 22-разрядных адресов.

Эти операции выполняются 12 конвейеризованными функциональными устройствами, которые могут работать параллельно во времени, выполняя различные операции. При этом возможно образование последовательных соединений конвейеров, в том числе так называемое *зацепление векторных операций*, при кото-

рых получаемые в ходе выполнения векторной команды результаты через регистры (без использования памяти) передаются для использования следующей векторной командой [16, 31]. Конвейеры имеют немного рабочих позиций (количество указано в скобках на рисунке), с тем чтобы уменьшить потери производительности, связанные с периодом «разгона» конвейера. Такт работы конвейера 12,5 нс.

Для поддержки конвейерной и векторной обработки необходимы *быстрые регистровые средства*, освобождающие от необходимости обращения к памяти при выполнении векторных операций и обеспечивающие высокий темп загрузки конвейеров и информационные связи между ними.

Процессор CRAY содержит быстрые регистры (время обращения 6 нс): 8 векторных *V*-регистров, каждый из которых может хранить 64 64-разрядных слова; 8 24-разрядных адресных регистров; 8 скалярных 64-разрядных *S*-регистров.

Управляющее устройство помимо традиционных для УУ блоков содержит четыре буфера команд (на 64 командных слова каждый) и специальные регистры: *регистр длины вектора* и *регистр векторной маски*. Регистр длины вектора фиксирует длину обрабатываемого вектора *S* (всегда  $S \leq 64$ ). Содержащий 64 разряда регистр векторной маски значением своего *i*-го разряда определяет, из какого из участвующих в операции регистров *i*-й элемент выдается в регистр результата. По-другому использует регистр векторной маски команда проверки элементов вектора, которая устанавливает разряды вектора маски в 1, если соответствующие элементы вектора удовлетворяют заданному условию. Результат этой команды может эффективно использоваться другими командами при обработке данных.

Команды имеют формат 16 или 32 разряда. 16-разрядные команды основных, в том числе векторных операций имеют 7-разрядное поле кода операции и три 3-разрядных поля для номеров регистров операндов и результата. В 32-разрядной команде под адрес основной памяти или непосредственный операнд отведено 22 разряда.

В памяти системы, имеющей емкость от 1 до 4 Мслов (слово 64 разряда + 9 контрольных), применено 16-кратное чередование адресов, за счет чего цикл обращения составляет всего 50 нс.

В однопроцессорной ВС CRAY-1 достигнута производительность около 80—130 Мфлоп/с. Для рассмотрения выбрана эта одна из первых конвейерно-векторных суперЭВМ из-за ее сравнительной простоты и наглядности в отношении особенностей организации комбинаций конвейерной и векторной обработок данных. В более новых и более мощных моделях фирмы CRAY



используются от 2 до 8 процессоров, и производительность составляет от 1000 до 2500 Мфлоп/с [64].

## 15.9. Концепция ВС с управлением потоком данных

Существуют трудности, связанные с решением проблемы автоматизации параллельного программирования, необходимой для обеспечения эффективного использования для широкого круга задач матричных (параллельных) ВС, т. е. систем типа ОКМД.

Конвейерные и конвейерно-векторные ВС наибольший эффект дают при реализации специализированных систем с фиксированными алгоритмами, при которых достижима оптимальная для решаемой задачи структура конвейера. В ВС универсального назначения состав позиций конвейера оказывается далеко не оптимальным для каждой данной задачи, к тому же возникают потери времени на коммутацию позиций конвейера под текущую команду.

В таких условиях актуальными являются исследования новых путей построения высокопроизводительных ВС, одними из которых являются *ВС с управлением потоком данных (операндов)*, или, другими словами, *потокковые ВС*.

В системах с управлением потоками данных предполагается наличие большого числа специализированных операционных блоков для определенных видов операций (сложения, умножения и т. п., отдельных для разных типов данных). Данные снабжаются указателями типа данных (*тегами*), на основании которых по мере готовности данных (операндов) к обработке (а не в порядке последовательности команд в программе) они загружаются в соответствующие свободные операционные блоки. При достаточном количестве операционных блоков может быть достигнут высокий уровень распараллеливания вычислительного процесса (близкий к «потенциальному параллелизму» программы). К тому же в самих операционных блоках может быть применена конвейерная обработка. Таким образом, создаются условия для реализации высокой производительности системы.

Во всех ранее рассмотренных машинах и вычислительных системах порядок выполнения операций над данными при решении задачи строго детерминирован, он однозначно определяется последовательностью команд программы.

Принципиальное отличие потокковых машин состоит в том, что команды выполняются не в порядке следования команд в тексте программы, а по мере готовности их операндов. Как только будут вычислены операнды команды, она может захватывать свободное операционное устройство и выполнять предпи-

Рис. 15.18. К примеру решения задачи на ВС с управлением потоком данных (граф потока операндов)

санную ей операцию. В этом случае последовательность, в которой выполняются команды, уже не является детерминированной.

Следуя [55], рассмотрим в качестве примера вычисление на потоковой ВС корней квадратного уравнения

$$ax^2 - bx + c = 0$$

(при  $b^2 - 4ac > 0$ );

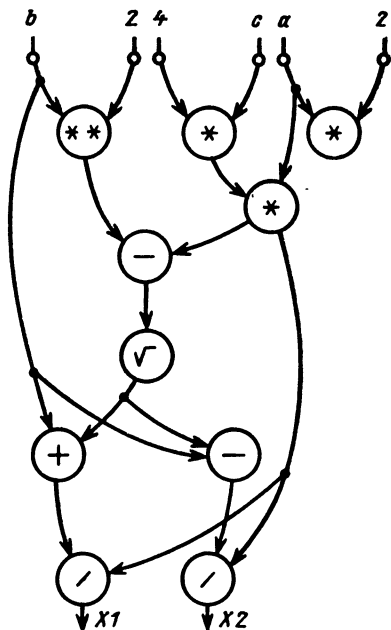
$$x_{1,2} = (b \pm \sqrt{b^2 - 4ac}) / 2a.$$

Полагаем, что на вход системы поступают группы данных  $a_i$ ,  $b_i$ ,  $c_i$  и по ним вычисляются корни уравнения. Решение может быть получено с помощью операционных устройств, выполняющих операции сложения, вычитания, умножения, возведения в степень, деления и извлечения квадратного корня.

Процесс решения можно представить следующей последовательностью операций:

- K0:A1:=2\*a;
- K1:B1:=b\*\*2 (возведение в степень);
- K2:C:=4\*c;
- K3:C1:=C\*a;
- K4:D:=B1-C1;
- K5:D1:=SQRT(D) (извлечение квадратного корня);
- K6:B2:=b+D1;
- K7:B3:=b-D1;
- K8:X1:=B2/A1;
- K9:X2:=B3/A1.

На рис. 15.18 показан граф потока данных, управляющих решением рассматриваемой задачи. Большими кружками обозначены операционные устройства, а маленькими — метки готовности данных. В начальном состоянии процесса готовы все входные данные. Вершины графа активизируются в произволь-



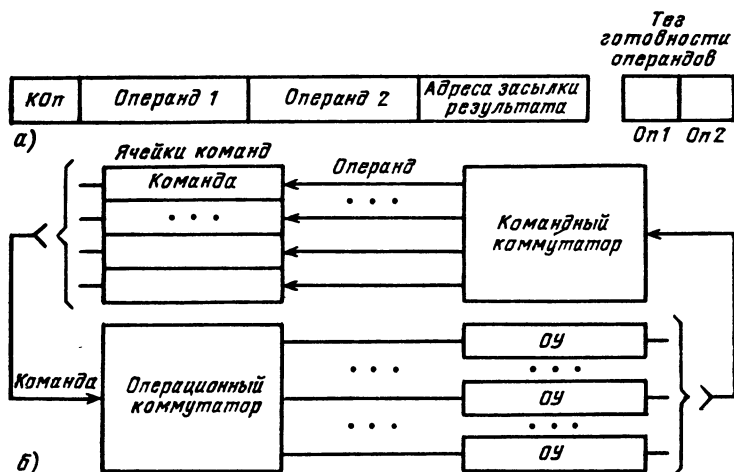


Рис. 15.19. Структура команды (а) и упрощенная структура процессора с управлением потоком данных (б):  
ОУ — операционное устройство

ном порядке при готовности их операндов. Тем не менее в конце концов получается один и тот же результат.

Можно считать, что активизация вершины сопровождается поглощением меток готовности на ее входах. По завершении операции меткой готовности отмечается выходная дуга вершины. Перемещение меток готовности по графу потока данных отображает протекание вычислительного процесса.

Возникает вопрос, как данные и соответствующие команды находят друг друга? Для ответа на него обратимся к рис. 15.19, поясняющему идею процессора, управляемого потоком данных [31, 34]. «Потоковая программа» размещается в массиве ячеек команд. Команда наряду с кодом операции содержит поля, куда заносятся готовые операнды, и поле, содержащее адреса команд, в которые должен быть направлен в качестве операнда результат операции. Кроме того, каждой команде поставлен в соответствие двухразрядный тег (располагаемый в управляющем устройстве), разряды которого устанавливаются в 1 при занесении в тело команды соответствующих операндов. В состоянии тега 11 (оба операнда готовы) инициируется запрос к операционному коммутатору на передачу готовой команды в соответствующее коду операции (и тегу операнда, определяющему тип данных) операционное устройство. Результат выполнения команды над ее непосредственно адресуемыми операнда-

ми направляется через командный коммутатор согласно указанным в команде адресам в ячейки команд и помещается в их поля операндов. Далее указанная процедура циклически повторяется, причем управление этим процессом полностью децентрализовано и не нуждается в счетчике команд.

### **Контрольные вопросы**

1. В чем различия в структурах и назначении многомашинных и многопроцессорных систем и комплексов?

2. Какой из комплексов — многомашинный или многопроцессорный — и почему предпочтительнее для использования при параллельной обработке потока задач: а) со слабыми информационными взаимосвязями; б) требующих интенсивного обмена промежуточными результатами в процессе решения задач?

3. Какие системные средства используются при организации в ЕС ЭВМ только в многомашинных и только в многопроцессорных ВК?

4. Что общего между двухканальным переключателем в ЕС ЭВМ и шинным переключателем в ЕС ЭВМ?

5. Какова природа конфликтов в многопроцессорных системах с межмодульными связями на основе общей (разделяемой) шины? Полностью ли исключает возникновение конфликтов коммутатор межмодульных связей?

6. Почему в системах с автоматической реконфигурацией предпочтительнее применение многопроцессорных структур?

7. Сравните по надежности и живучести структуры отказоустойчивых комплексов, описанных в § 15.4?

8. Какие подходы и структуры используются при построении супер-ЭВМ?

9. Как данные (операнды) и соответствующие команды находят друг друга в ВС с управлением потоком данных?

## *Глава 16*

# **ПРИНЦИПЫ ОРГАНИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ СЕТЕЙ**

## **16.1. Вычислительные сети. Общие сведения**

Развитие архитектуры, аппаратурных и программных средств ВС и успехи в развитии методов организации и аппаратуры систем передачи данных по каналам связи позволили приступить к созданию вычислительных систем качественно нового типа — вычислительных сетей.

*Вычислительной сетью* (ВСт) или *сетью ЭВМ* называется комплекс территориально рассредоточенных ЭВМ и терминальных устройств, связанных между собой каналами передачи данных.

Целесообразность создания ВСт обуславливается возможностью использования территориально рассредоточенными пользователями оборудования ЭВМ, программ и информационных баз, находящихся в различных вычислительных центрах сети, возможностью организации «распределенной обработки» данных путем привлечения вычислительных ресурсов нескольких вычислительных центров сети для решения особо сложных задач.

Вычислительную сеть можно рассматривать как систему с распределенным по территории аппаратными, программными и информационными ресурсами. Возможна реализация на основе ВСт распределенного (децентрализованного) *банка данных*, отдельные информационные базы которого создаются в местных вычислительных центрах, например, в процессе функционирования АСУ отдельных предприятий и объединений, а при решении задач более высокого уровня управления используются как единая база данных.

Другая возможность — это создание *централизованного банка данных*, к которому имеют доступ многочисленные, в том числе находящиеся на значительном расстоянии абоненты через свои терминалы (*абонентские пункты*) и терминалы местных систем коллективного пользования.

Объединение в сеть ЭВМ нескольких вычислительных центров способствует повышению надежности функционирования вычислительных средств, так как создается возможность резервирования одних вычислительных центров за счет технических ресурсов других центров.

Вычислительная сеть позволяет оперативно перераспределять нагрузку между ЭВМ сети и снижать пиковую нагрузку на вычислительные средства.

С созданием ВСт возникли предпосылки для специализации отдельных ВЦ сети на решении задач определенного класса, что по оценкам специалистов дает значительный эффект, так как позволяет существенно сократить общие затраты высококвалифицированного труда на разработку моделей, алгоритмов и пакетов прикладных программ. Специализация отдельных ВЦ сети становится возможной, так как пользователь имеет доступ к уникальным программам и данным, а также уникальным вычислительным средствам (например, специализированным процессорам) любого ВЦ сети.

В ВСт с программно-несовместимыми ЭВМ теряется острота проблема переноса программных средств с одних машин на

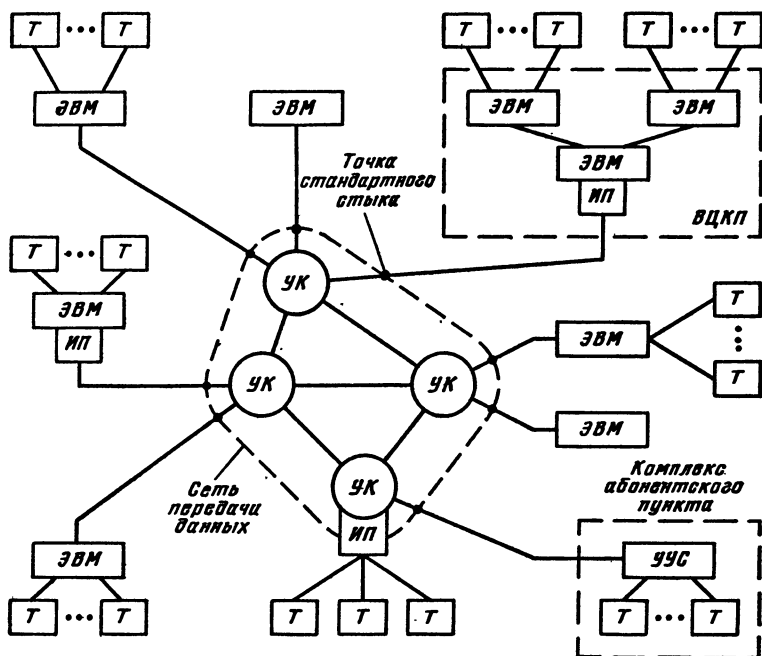


Рис. 16.1. Структурная схема вычислительной сети

другие, так как пользователь может воспользоваться именно той машиной, для которой нужна программа имеется.

Вычислительные сети позволяют решать качественно новые задачи, такие, например, как создание распределенных по обширной территории систем информационного обслуживания научных и конструкторских работ, автоматизированных обучающих систем или систем резервирования билетов на различные виды транспорта.

Обобщенная структурная схема ВСт приведена на рис. 16.1. Основу ВСт составляют крупные ЭВМ (вычислительные центры коллективного пользования — ВЦКП), объединяемые сетью передачи данных. Эти ЭВМ, называемые *главными вычислительными машинами* (ГВМ), выполняют основные функции по выполнению программ пользователей, сбору, хранению, выдаче информации. Сеть передачи данных (СПД) образуют каналы связи и узлы (центры) коммутации (УК), в которых сетевые (сетевые) процессоры (СП) управляют выбором маршрутов передачи данных в сети, выполняют функции мультиплексоров,

концентратора каналов, осуществляют коммутацию каналов, сообщений или пакетов.

Главные вычислительные машины подсоединяются к сети (точнее, к СП) непосредственно через точки стандартного стыка, если обеспечена совместимость по физическим сигналам (управляющим и информационным) и форматам информации между ГВМ и СП, или с помощью интерфейсных процессоров (ИП). В последнем случае интерфейсный процессор (обычно это микроЭВМ) может выполнять и некоторые функции по предварительной обработке данных [преобразование кодов передаваемых (принимаемых) данных, контроль правильности полученных сообщений и др.], разгружая от этих операций ГВМ.

Терминалы (Т) пользователей подключаются либо к ГВМ (ВЦКП), либо непосредственно к СП. Для подключения к сети группы терминалов, удаленных от ГВМ и СП, используются терминальные процессоры (концентраторы), называемые часто *абонентскими пунктами* (АП). В этом случае отпадает необходимость выделения каждому терминалу отдельного канала связи.

Абонентский пункт содержит устройство управления и связи (УУС). В качестве терминалов используются телетайпы, электрифицированные пишущие машинки, дисплеи, графопостроители и другие устройства ввода-вывода, а также их комбинации. Оборудование терминала может включать в себя микроЭВМ, при этом терминал будет выполнять некоторые функции по вспомогательной обработке информации, что служит основанием называть его *«интеллектуальным терминалом»*.

Абонентский пункт содержит устройство управления и связи (УУС).

Интегрируемые в вычислительную сеть ГВМ, обычно сохраняющие в основном функции по отношению к своим вычислительным центрам, берут на себя часть общесетевой нагрузки по сбору, обработке и передаче данных.

Административное управление в ВСт включает в себя планирование и учет работы отдельных машин сети, анализ и учет работы сети передачи данных, проведение измерений на сети («сетиметрия») и т. п. Эти функции возлагают на одну из ГВМ сети, которую называют административным комплексом.

Отмеченные выше направления использования ВСт поддерживаются реализуемыми в сетях специфическими режимами работы, позволяющими осуществлять:

- а) обмен сообщениями между терминалами;
- б) удаленный ввод заданий с любого терминала через сеть на выполнение пакетной или диалоговой обработки на удаленной ЭВМ;

в) доступ к удаленным файлам и передача файлов между ЭВМ сети.

За рубежом и в Советском Союзе создан ряд крупных ВСт, ведется интенсивная разработка новых сетей. Некоторые из существующих и разрабатываемых ВСт охватывают обширные территории, имеют общегосударственный и даже межгосударственный масштаб функционирования.

Одной из крупнейших является вычислительная сеть АРПА (США), объединяющая свыше 300 программно-несовместимых ЭВМ разной производительности. Вычислительная сеть АРПА охватывает значительную часть территории США и через спутниковую связь связана с несколькими вычислительными центрами в Европе.

В Советском Союзе функционирует ряд отраслевых вычислительных сетей, экспериментальная ВСт Академии наук СССР, создана межгосударственная, охватывающая страны СЭВ сеть научно-технической информации.

*Протоколы вычислительных сетей.* Управление таким сложным процессом, как передача данных в ВСт, в котором участвует многочисленная и разнообразная аппаратура, требует формализации и стандартизации процедур установления и разъединения соединений, передачи информации, контроля правильности передачи, исправления ошибок, выделения и освобождения ресурсов ЭВМ и сети передачи данных. Эти задачи решаются с помощью протоколов, регламентирующих стандартизованные процедуры взаимодействия элементов сети при установлении связи и передаче данных. Реализацией протокольных процедур управляют специальные программы или аппаратурные средства.

Проводится интенсивная работа по стандартизации протоколов ВСт в международном масштабе. Основной организацией по разработке международных стандартов в области связи является Международный консультативный комитет по телеграфии и телефонии (МККТТ), который по вопросам, относящимся к ВСт, действует совместно с Международной организацией по стандартизации (ИСО), разрабатывающей стандарты в области ЭВМ и обработки данных. Стандартизацией протоколов локальных сетей активно занимается комитет 802 Института инженеров по электротехнике и радиоэлектронике (IEEE) в США.

## 16.2. Классификация вычислительных сетей

Классификацию ВСт можно производить по ряду признаков.

По функциональному назначению различают сети *информационные*, представляющие пользователю в основном информа-



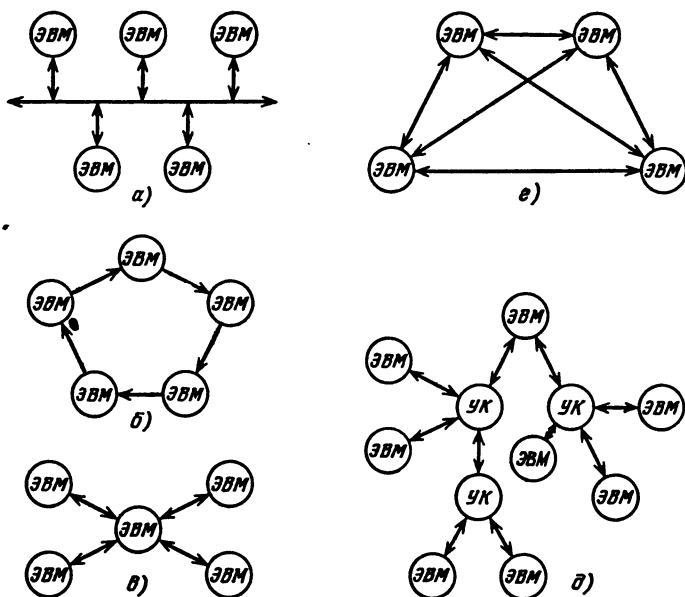


Рис. 16.2. Виды топологий вычислительных сетей:

а — одинарная многоточечная линия; б — кольцевая (петлевая) сеть; в — звездообразная сеть; г — полностью связанная сеть; д — древовидная сеть

ционное обслуживание (сети научно-технической информации, здравоохранения, резервирования билетов на транспорте и т. п.), *вычислительные*, выполняющие главным образом решение задач с обменом данными и программами между ЭВМ сети, и смешанные — *информационно-вычислительные*.

По размещению информации в сети разделяют сети с централизованным банком данных, формируемым в одном из узлов сети, и с распределенным банком данных, состоящим из отдельных локальных банков, расположенных в узлах сети.

По степени территориальной рассредоточенности можно выделить крупномасштабные или *глобальные* ВСт, охватывающие территорию страны, нескольких стран с расстояниями между отдельными узлами сети, измеряемыми тысячами километров, *региональные сети*, охватывающие определенные территориальные регионы — город, район, область и т. п., *локальные вычислительные сети* (ЛВСт) с максимальным расстоянием между узлами сети не более нескольких километров.

По числу ГВМ, в которых производятся сбор, хранение

и обработка информации, следует различать ВСт с несколькими ГВМ и с одной. К последним относятся упоминавшиеся в гл. 1 ВС с телеобработкой, которые представляют собой комплексы, состоящие из вычислительной машины и удаленных абонентских пунктов, связанных с помощью каналов и аппаратуры передачи данных.

По типу используемых ЭВМ выделяют *однородные сети*, содержащие программно-совместимые машины, и *неоднородные*, если машины сети программно-несовместимы. На практике сети часто являются неоднородными.

По методу передачи данных различают вычислительные сети с *коммутацией каналов*, с *коммутацией сообщений*, с *коммутацией пакетов* и со смешанной коммутацией. Для современных ВСт характерно использование коммутации пакетов (см. § 16.4).

Признаком различия сетей является тип используемых в ней протоколов обмена информацией.

Важным признаком классификации ВСт является их *топология*, определяющая геометрическое расположение основных ресурсов вычислительной сети и связей между ними. Топологическая структура ВСт оказывает значительное влияние на ее пропускную способность, устойчивость сети к отказам ее оборудования, на логические возможности и стоимость сети. В настоящее время наблюдается большое разнообразие в топологических структурах вычислительных сетей.

На рис. 16.2 представлены различные варианты топологий ВСт. Топология крупных вычислительных сетей может представлять собой комбинацию нескольких топологических решений.

### **16.3. Методы передачи данных по каналам связи. Коммутация каналов, сообщений, пакетов**

Каналом связи называют физическую среду и аппаратные средства, осуществляющие передачу информации от одного узла коммутации к другому либо к абоненту связи. Под физической средой понимается пространство или материал, обеспечивающие распространение сигналов: проводная воздушная или кабельная линия, скрученная пара проводов, коаксиальный кабель, световодная (стекловолоконная) линия, эфир [69].

В технике связи используются телеграфные и телефонные каналы связи. По телеграфным каналам информация передается в дискретной форме, что облегчает их сопряжение с ЭВМ. Однако скорость передачи информации по телеграфным каналам невелика и составляет всего 50—200 бит/с. По телефонным каналам информация (речь) передается в аналоговой форме, и поэтому значительно усложняется сопряжение этих каналов

с ЭВМ. Однако эти каналы позволяют при использовании соответствующей дополнительной аппаратуры производить передачу двоично-кодированной информации с достаточно большей скоростью (например, 96 000 бит/с).

В настоящее время в технике глобальных и региональных вычислительных сетей и телеобработки для передачи данных используют главным образом аналоговые каналы связи. При этом требуется специальная аппаратура передачи данных (АПД), осуществляющая необходимые для сопряжения аналоговых каналов с ЭВМ преобразования форм представления передаваемой информации, а также некоторые другие операции.

Канал связи, оснащенный аппаратурой для передачи дискретной информации, называется *каналом передачи данных*.

По возможным направлениям передачи информации различают каналы:

- 1) *симплексные*, позволяющие передавать данные только в одном направлении;
- 2) *полудуплексные*, передающие данные в обоих направлениях, но не одновременно;
- 3) *дуплексные*, позволяющие передавать одновременно данные в обоих направлениях.

Каналы передачи данных в зависимости от скорости передачи делятся на низкоскоростные со скоростью передачи 50, 75, 100, 200 бит/с (по телеграфным линиям связи), среднескоростные со скоростью передачи 600, 1200, 2400, 4800 и 9600 бит/с (по телефонным каналам), высокоскоростные со скоростью 24 000, 48 000, 96 000 бит/с (по ВЧ-каналам).

Передача дискретной (двоично-кодированной) информации по аналоговым каналам связи осуществляется путем *модуляции* в передающем пункте колебаний несущей частоты (значение частоты зависит от скорости передачи данных; наименьшая из употребляемых частот 1500 Гц) двоичными сигналами передаваемого кода с последующей *демодуляцией* — восстановлением первоначальной формы сообще-

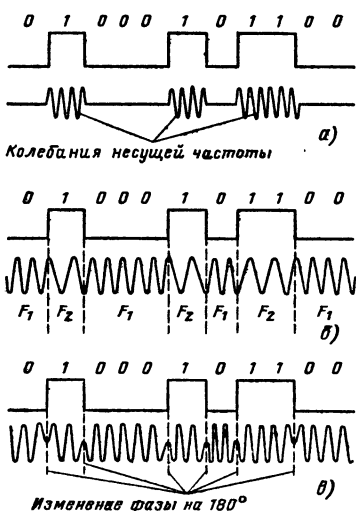


Рис. 16.3. Виды модуляции:  
а — амплитудная; б — частотная;  
в — фазовая



Рис. 16.4. Дискретный канал передачи данных

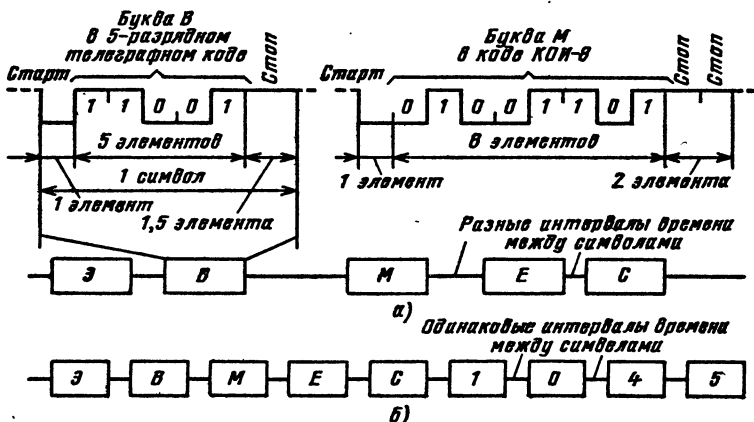


Рис. 16.5. Асинхронная (а) и синхронная (б) передачи данных

ния в приемном пункте. Операции модуляции и демодуляции выполняются в устройствах, называемых *модемами*.

Применяются три основных вида модуляции: *амплитудная, частотная и фазовая*, которые поясняются на рис. 16.3. Способ образования дискретного канала передачи данных показан на рис. 16.4.

Сама передача данных в канале связи может быть асинхронной и синхронной (рис. 16.5). При *асинхронной передаче* символы передаются в свободном темпе независимо друг от друга, причем каждый символ передается со своими сигналами *Старт* и *Стоп*, указывающими на начало и конец передачи символа.

При *синхронной передаче* блок символов передается непрерывно в принудительном темпе, синхронизация передающего и принимающего устройств достигается посылкой специальных кодовых комбинаций перед каждым блоком данных.

Асинхронная передача позволяет передавать информацию с устройств, которые выдают ее асинхронно во времени (например, ручные клавиатуры печатающих машинок, дисплеев и т. д.).

Однако скорость передачи информации при асинхронном методе низка, так как велика ее избыточность из-за большого числа служебных сигналов. Поэтому асинхронная передача используется только при скорости передачи до 200 бит/с. Синхронный метод обеспечивает большую скорость передачи данных из-за меньшей избыточности информации, но требует более сложной аппаратуры.

Помехи в каналах связи могут вызывать ошибки при передаче информации. *Достоверность передачи данных* оценивается отношением числа ошибочно принятых символов к общему числу переданных. Для телефонных коммутируемых каналов достоверность передачи составляет  $10^{-3}$ . Такое низкое значение достоверности передачи заставляет в ряде случаев применять специальные методы (контроль по четности, контрольные суммы, циклические коды) и средства контроля правильности передачи, автоматического повторения передачи при появлении ошибки или автоматической коррекции.

Большую часть стоимости вычислительных сетей и крупных систем телеобработки составляют каналы связи. Это делает необходимым применение уплотнения и концентрации каналов связи.

*Уплотнение каналов* связи представляет собой статическое разделение канала, при котором определенные полосы частот или периоды времени в уплотняемом канале выделяются в фиксированном, заранее заданном порядке для использования в качестве отдельных каналов.

*Концентрация каналов связи* в отличие от уплотнения является динамической процедурой распределения меньшего числа более скоростных выходных каналов концентратора в соответствии с имеющимися запросами между большим числом менее скоростных входных каналов.

*Передача данных* в ВСт производится следующими методами: коммутацией каналов, коммутацией сообщений и коммутацией пакетов (рис. 16.6) [30]. Выбор метода зависит от назначения сети и характера передаваемых данных.

*Коммутация каналов.* При этом методе в сети передачи данных устанавливается физическое соединение между пунктами отправления и назначения (источником и адресатом) путем образования составного канала из последовательно соединенных отдельных канальных участков. Установление связи между источником и адресатом производится путем послыки пунктом отправления сигнализирующего сообщения, которое, перемещаясь по сети передачи данных от одного узла коммутации каналов к другому и занимая пройденные каналы, прокладывает путь от источника к пункту назначения. Этот путь (составной



канал) состоит из физических каналов, имеющих одну и ту же скорость передачи данных. Об установлении физического соединения из пункта назначения в источник посылается *сигнал обратной связи*. Затем из источника передается сообщение по установленному пути с одновременным использованием всех образующих его каналов, которые оказываются недоступными для других передач, пока источник их не освободит.

При коммутации сообщений и пакетов информация передается с запоминанием в промежуточных узлах сети передачи данных без установления физического соединения между пунктами отправления и назначения. Между ними устанавливается *виртуальное или логическое соединение*.

*Коммутация сообщений.* При этом методе физическое соединение устанавливается только между соседними узлами сети (называемыми центрами или узлами коммутации сообщений) и только на время передачи сообщения. Каждое сообщение снабжается заголовком и транспортируется по сети как единое целое. Поступившее в узел сообщение запоминается в его буферном ЗУ и в подходящий момент, когда освободится соответствующий канал связи, передается в следующий, соседний узел. Сообщение как бы прыгает от одного узла к другому, занимая в каждый момент передачи только канал между соседними узлами, при этом виртуальный канал между источником и адресатом может состоять из физических каналов с разной скоростью передачи данных. Коммутация сообщений по сравнению с коммутацией каналов позволяет ценой усложнения аппаратуры узла коммутации уменьшить задержку при передаче данных и повысить общую пропускную способность сети передачи данных.

*Коммутация пакетов* является развитием метода коммутации сообщений. Она позволяет добиться дальнейшего увеличения пропускной способности сети, скорости и надежности передачи данных.

Поступающее от абонента сообщение подвергается в интерфейсных процессорах пакетированию — разбивается на пакеты, имеющие фиксированную длину, например 1 Кбит (рис. 16.7). Пакеты метятся служебной информацией — заголовком, указывающим адрес пункта отправления, адрес пункта назначения и номер пакета в сообщении. Пакеты транспортируются в сети как независимые сообщения и поступают в узел коммутации пакетов, где накапливаются в буферах каналов связи. Затем они передаются в выходной буфер, где попеременно накапливаются пакеты различных сообщений, которые выдаются на скоростной канал связи для передачи в соседний УК. В пункте назначения интерфейсный процессор формирует из пакетов исходное сообщение.

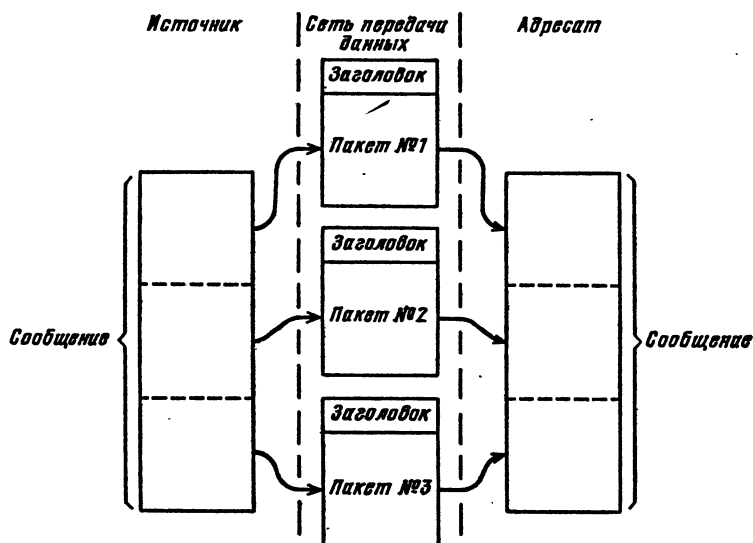


Рис. 16.7. Пакетирование сообщений

Важным достоинством коммутации пакетов является возможность одновременной передачи пакетов одного и того же сообщения разными маршрутами, что уменьшает время и увеличивает надежность передачи сообщения.

При передаче короткими пакетами уменьшаются вероятность появления ошибок и время занятости каналов повторными передачами.

Как видно из рис. 16.6, при коммутации каналов имеется наибольшая задержка в передаче информации, что определяется сравнительно большим временем установления связи из-за ожидания освобождения промежуточных каналов. Монополизация источником на все время связи каналов, образующих путь от него к адресату, снижает общую пропускную способность сети передачи данных. Однако коммутация каналов обладает важной особенностью: если связь источник — адресат установлена, то при передаче сохраняются неизменными временные соотношения между элементами передаваемой информации (*временная прозрачность*), что имеет значение для систем, работающих в реальном масштабе времени. Если в таких системах недопустимы задержки с установлением связи, то используют сравнительно дорогие некоммутируемые (постоянно выделенные) каналы связи.

Метод коммутации пакетов по сравнению с другими методами обеспечивает наименьшую задержку при передаче данных



и наибольшую пропускную способность сети передачи данных, особенно заметную при передаче коротких сообщений, характерных для диалогового режима. Использование коммутации пакетов способствует повышению надежности и живучести сети вследствие того, что облегчается адаптация управления передачей данных к отказам и перегрузкам отдельных участков. Поэтому в настоящее время коммутация пакетов является основным методом передачи данных в ВСт, но во многих случаях этот метод оказывается непригодным для систем, работающих в реальном времени.

В ВСт используют два характерных режима передачи пакетов: режим виртуальных каналов и режим дейтаграмм.

*Режим виртуальных каналов* предполагает, что передаче сообщения предшествует прокладка (организация) *виртуального канала (логического соединения)*, по которому затем строго в порядке номеров передаются пакеты сообщения. Такой режим облегчает сборку пакетов в сообщение на приемном конце.

*Режим дейтаграмм* допускает независимое перемещение по сети пакетов сообщения (называемых в этом случае *дейтаграммами*) и не требует предварительного установления логического соединения. Этот режим более сложен в реализации, так как усложняется сборка пакетов в сообщение у адресата, однако позволяет достигнуть большей надежности и живучести сети передачи данных.

Для выбора маршрута пакета в сети могут использоваться статические процедуры, основанные на наличии таблиц предпочтительных маршрутов в УК, и различные варианты адаптивной процедуры. В отказоустойчивых сетях часто используется лавинный метод, согласно которому пакет посылается из каждого исходного узла во все соседние. Если пакет попадает в узел, где он уже был, то пакет уничтожается.

#### **16.4. Эталонная логическая модель вычислительной сети и иерархия протоколов**

Сложность и разнообразие реальных физических структур вычислительных сетей, разнотипность используемых в вычислительных сетях ЭВМ и другой аппаратуры, необходимость упорядочения разработки программных и аппаратурных средств сетевого комплексирования вычислительных установок делают целесообразным введение обобщенной логической структуры вычислительной сети, с которой могут быть соотнесены физические структуры конкретных сетей.

Такая обобщенная логическая структура предложена Международной организацией по стандартизации (ИСО) в виде *эта-*

лонной семиуровневой логической модели открытых систем [71]. Системы называются открытыми друг для друга в том смысле, что, несмотря на их технические и логические различия, они могут взаимодействовать с помощью определенных процедур.

В вычислительной сети связь в действительности устанавливается не между отдельными машинами, а между прикладными программами, или, более точно, между *прикладными вычислительными процессами*, протекающими в машинах. Под прикладным процессом следует понимать прикладную программу вместе с ее наборами данных и выделенными ей ресурсами машины.

Процесс можно рассматривать как *логический (виртуальный) процессор*. Он получает сообщения от других процессов, выполняемых в этой же или в других ЭВМ, и выдает сообщения другим процессам через свои программно-реализуемые логические входные и выходные порты.

Связь между процессами реализуется с помощью целого ряда специальных процессов и обслуживающих их аппаратурных и программных средств, например процессов установления маршрута передачи сообщения, управления передачей, установления соединения, интерпретации принятого сообщения и др. Это обстоятельство закономерно приводит к рассмотрению иерархии функций, реализуемых в оборудовании сети.

В рассматриваемой модели под системой понимается иерархически упорядоченная совокупность функций, обеспечивающая выполнение некоторым аппаратурным комплексом, включающим в себя одну или несколько ЭВМ, приписанного ему в сети функционального назначения.

По функциональному назначению различают *абонентские системы* (ГВМ, терминальные комплексы, ЭВМ — администраторы сети) и *коммутационные системы*, участвующие в управлении передачей информации в сети.

Связи между системами в логической модели называются физическими соединениями. Для упорядочения функций системы вводятся функциональные уровни (рис. 16.8).

В общем случае абонентская система имеет следующие семь функциональных уровней, перечисленных ниже в порядке возрастания уровня иерархии:

1. Обеспечение электрических, механических и функциональных характеристик подключения к физическим каналам связи, преобразование сигналов — *физический уровень* или *уровень управления физическим каналом*.

2. Управление каналом передачи данных, установление, поддержание и разъединение каналов (соединений), защита от ошибок при передаче данных — *канальный уровень* или *уровень управления информационным каналом*.

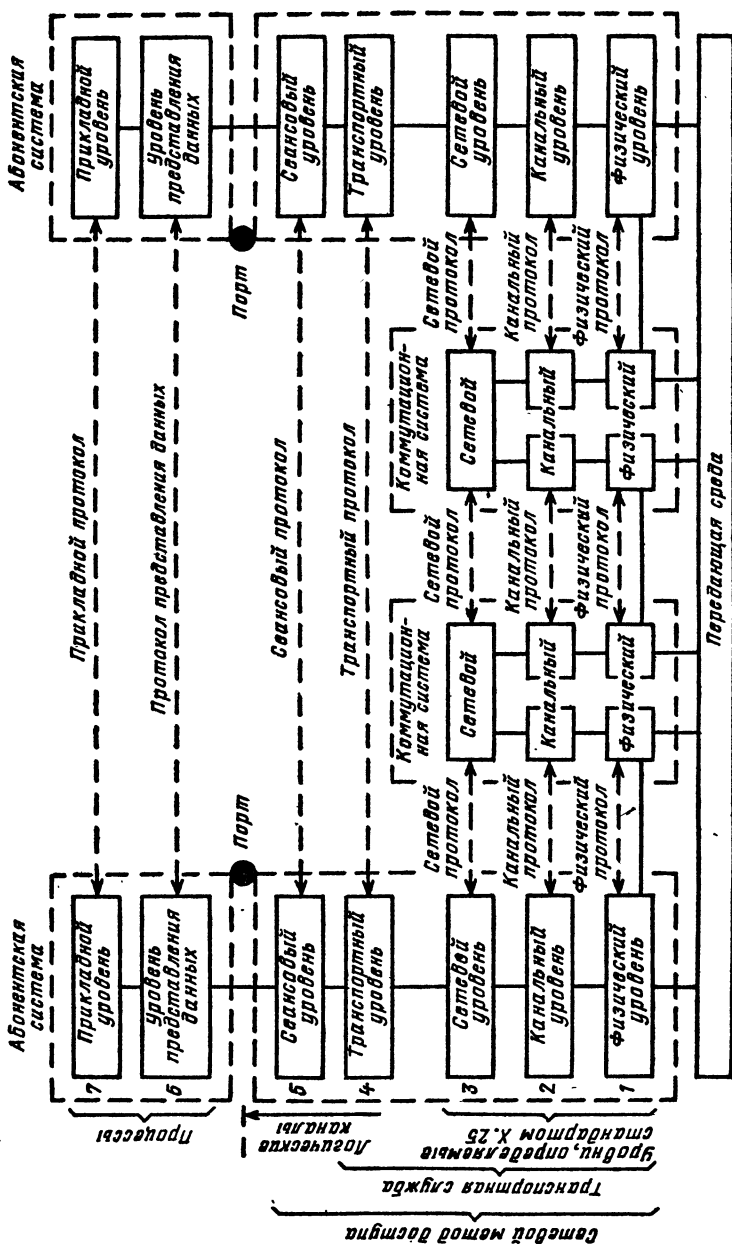


Рис. 16.8. Эталонная семнуровневая логическая модель вычислительной сети. Иерархия протоколов сети

3. Маршрутизация, коммутация и адресация информации, управление потоками данных (*организация виртуальных каналов* между абонентами и передачи по ним пакетов) — сетевой уровень.

4. Управление передачей данных (без обработки их в промежуточных узлах) от системы-источника к системе-адресату — *транспортный уровень*.

5. Организация и проведение сеансов связи между прикладными процессами — *сеансовый уровень*.

6. Интерпретация и преобразование передаваемых между процессами данных к виду, удобному для прикладных процессов, — *уровень представления данных*.

7. Выполнение прикладных программ, управление терминалами, административное управление сетью — *прикладной уровень*.

Первые четыре уровня образуют *транспортную службу* вычислительной сети, освобождающую более высокие уровни от забот по организации передачи данных. Части вычислительной сети, реализующей транспортную службу, в логической структуре ВСт соответствует *транспортная сеть*.

Функции коммутационной системы обычно ограничиваются тремя нижними уровнями рассматриваемой модели.

Уровню могут быть поставлены в соответствие некоторые процессы, аппаратурные и программные средства (*объекты уровня*).

Каждый уровень обслуживает соседний старший уровень. Связь между объектами соседних уровней одной системы регламентируется *межуровневым интерфейсом*. Организация взаимодействия между одинаковыми уровнями различных систем определяется соответствующим *протоколом*.

Введенная выше иерархия уровней функций логической системы определяет соответствующую иерархию протоколов и соответственно протокольных программных или аппаратурно-программных средств, действующих в вычислительной сети, как это показано на рис. 16.8.

Два старших уровня (6 и 7) соответствуют *процессам* (процессам представления и преобразования данных, выполнения прикладных программ, административного управления сетью). Остальные уровни в совокупности определяют *сетевой метод доступа* к указанным процессам, осуществляемый через порты процессов.

Управление взаимодействием различных объектов сети осуществляется с помощью сетевого программного обеспечения — *транспортных станций*, представляющих собой специальные программы, располагаемые в ГВМ, а также *сетевых программ* связных процессоров. Транспортная станция обеспечивает мульт-

типлексирование независимых потоков данных от многих абонентов. Она может быть реализована как часть операционной системы ЭВМ либо как независимая системная задача.

Нижние три уровня и соответственно физический, канальный и сетевой протоколы охватывает международный стандарт-протокол X.25; предусматривающий передачу данных путем коммутации пакетов, при этом разборка сообщения на пакеты при передаче и сборка пакетов в сообщение при приеме производятся в ГВМ или в интерфейсных процессорах.

Протоколы уровней 4—7 («сквозные») регламентируют процедуры «сквозной» связи между абонентами сети.

## 16.5. Элементы протоколов

Протоколы разных уровней управления, реализуемые программными, аппаратными, а в ряде случаев и специальными командными средствами, представляют собой формализованные процедуры взаимодействия процессов одного уровня территориально удаленных систем, связанных сетью передачи данных. Сложность протоколов делает целесообразным выделение некоторых их типичных элементов.

*Обрамление (конвертование) сообщений.* Передаваемое сообщение, сформировавшееся на верхнем (прикладном) уровне управления, последовательно поступает на более низкие уровни управления своей системы, затем по физическому каналу, пройдя промежуточные коммутационные системы, передается в приемную систему, где последовательно проходит с более низких уровней управления на более высокие, вплоть до прикладного.

В системе-источнике (ЭВМ-А) подлежащие пересылке данные на каждом уровне обрамляются специфической служебной информацией: *заголовком*, а в некоторых случаях и *концевиком*, содержащими идентификаторы (адреса) приемника и источника сообщения, портов, тип сообщения, номера логического канала, номера пакета (кадра), контрольные коды и др. Оформленные таким образом данные назовем *контейнером*. Контейнеры, сформированные на разных уровнях управления, имеют свои собственные названия (рис. 16.9).

Прикладной, представительный и сеансовый уровни совместно формируют из подлежащего передаче сообщения контейнеры, называемые *блоками* (в ряде случаев в блоке отсутствует концевик процесса). Транспортный уровень формирует из блока контейнер *фрагмент*. Сетевой уровень формирует *пакет*. Канальный уровень формирует *кадр*, который и передается через физический канал в виде последовательности бит.

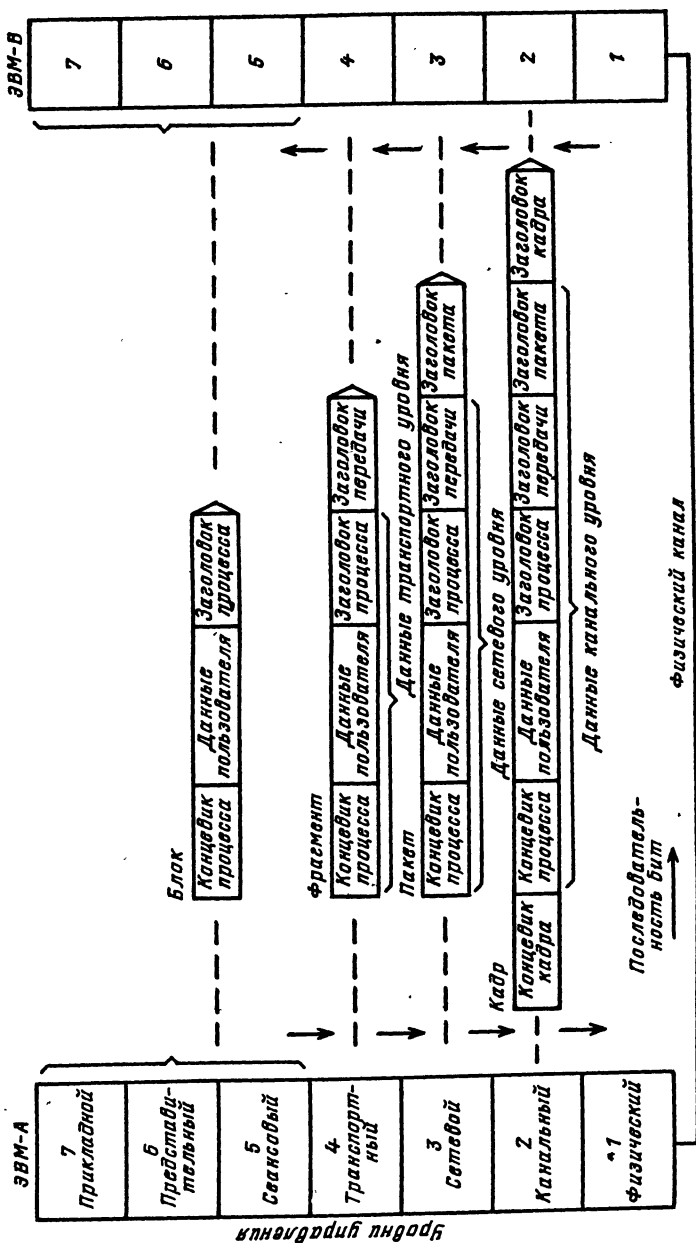


Рис. 16.9. Процедура оформления сообщения. Структура блока, фрагмента, пакета, кадра

Формирование контейнеров подобно многократному конвертованию писем с формированием соответствующих надписей на конвертах.

В приемной системе, в которой контейнеры перемещаются с более низких уровней управления на более высокие, последовательно используется и отбрасывается информация, заключенная в заголовках и концевиках, т. е. происходит распаковка контейнеров (расконвертование), и данные достигают прикладного уровня адресованной системы.

В промежуточной коммутационной системе принятый ею кадр после обработки на канальном уровне (проверки правильности передачи) поступает в форме пакета на ее сетевой уровень, определяющий направление его дальнейшего движения. Затем пакет передается на канальный уровень, где оформляется кадр, отсылаемый коммутационной системой.

На каждом уровне используется только соответствующий этому уровню заголовок контейнера (и концевик), а данные этого уровня без каких-либо изменений передаются на следующий, более высокий уровень.

*«Рукопожатие»* — элементарная протокольная процедура, приписывающая обмен определенными управляющими словами (командами) или ответами (регламентированными сообщениями) между передающей и принимающей системами при установлении соединения, выполнении сеанса передачи данных и разъединении соединения.

*Тайм-аут* — предельный промежуток времени ожидания получения «квитанции» о правильном приеме переданного сообщения. Неполучение подтверждения за время тайм-аута вызывает повторную передачу.

*«Окно»*. В целях повышения пропускной способности сети обычно нецелесообразно откладывать посылку следующего кадра (пакета) до прихода подтверждения о получении адресатом предыдущего. В таком случае используется механизм «окна», устанавливающий в зависимости от условий, складывающихся при работе сети передачи данных, «ширину окна», т. е. предельную допустимую разность количеств переданных и полученных кадров (пакетов).

*Прозрачность* — свойство протокола освобождать протоколы более верхнего уровня от необходимости в какой-либо форме учитывать специфические особенности процедур данного протокола, полностью освобождать протоколы верхних уровней по отношению к рассматриваемому уровню от каких-либо действий, связанных с учетом этих специфических процедур. Прозрачность протокола некоторого уровня проявляется в том, что он пропускает через себя без каких-либо искажений и изменений дан-

ные, команды и другую информацию, сформированную на более высоких уровнях управления.

В качестве примера возникновения проблемы прозрачности протокола и способа ее обеспечения обратимся к канальному протоколу «Высокоуровневое управление каналом передачи данных» (HDLC), согласно которому кадр с обеих сторон снабжается 8-битными флагами, отделяющими кадры друг от друга в потоке передаваемых бит. Флаг задается кодом 01111110, т. е. шестью единицами, окаймленными нулями.

Нарушение прозрачности протокола могло бы возникнуть из-за того, что среди передаваемых данных может встретиться комбинация бит, соответствующая флагу, которая ошибочно может быть воспринята на канальном уровне как начало или конец некоторого кадра. Если не придумать способ разрешения этой «непрозрачности», протоколу верхнего уровня придется каким-то образом преобразовывать передаваемые данные, искусственно устраняя нежелательные комбинации бит.

Однако заботу об устранении этого затруднения берет на себя сам канальный протокол (тем самым обеспечивая свою прозрачность для потока передаваемых бит) с помощью процедуры, получившей название «бит-стаффинг».

Эта процедура предусматривает, что передающая сторона в случае появления при передаче данных (не флагов) последовательности из пяти единиц производит вставку дополнительного нулевого бита, а приемная сторона в случае появления в потоке бит последовательности из пяти единиц изымает дополнительный нулевой бит и полностью восстанавливает исходный вид сообщения.

*Виртуализация устройств (процессов) сети.* Для обеспечения организации обмена информацией в ВСт с характерным для них большим разнообразием типов входящих в их состав ЭВМ и используемых ими операционных систем, терминалов и другого оборудования, а также разнообразием применяемых в компонентах сети форм представления данных и управляющей информации в сетях вводятся и четко определяются понятия *виртуальное задание, виртуальный терминал, виртуальный файл*.

Иницилирующая удаленную обработку абонентская система (ЭВМ или терминал) в принятой для данного оборудования форме генерирует задание, которое ее представительный уровень управления преобразует в принятую в сети форму *виртуального задания*. В такой форме задание через сеть передается в систему-исполнитель и там в результате действия ее представительного уровня приобретает используемую в этой системе форму задания.

Подобным же образом при передаче файла в передающей



*Интерфейс  
высокого уровня*

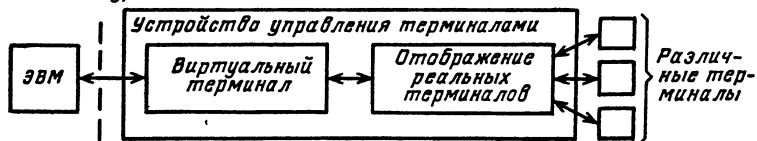


Рис. 16.10. Принцип действия виртуального терминала

абонентской системе он преобразуется в *виртуальный файл*, который, поступив в принятую систему, преобразуется в принятую в ней форму представления файлов.

*Виртуальный терминал* — это гипотетический обобщенный терминал, реализуемый в виде входящего в состав устройства управления реальным терминалом программного или программно-аппаратурного средства, позволяющего отображать реальные терминалы на виртуальный терминал. Абонентские системы имеют дело с виртуальными терминалами, которым приписывается набор некоторых универсальных процедур. Виртуальный терминал использует интерфейс высокого уровня при взаимодействии через сеть с ЭВМ, а для связи с реальным терминалом специализированный интерфейс, основу которого составляет процедура отображения реального терминала (рис. 16.10) [37].

## 16.6. Протоколы управления физическим и информационным каналами и сетью передачи данных. Протокол X.25

Принятый МККТТ и ИСО протокол (рекомендация) X.25 фактически включает в себя протоколы трех нижних уровней рассмотренной в § 16.4 эталонной логической семиуровневой модели вычислительной сети: на физическом уровне — стандартный интерфейс X.21, на канальном уровне — протокол управления информационным каналом X.25/2, в качестве которого используется «процедура доступа к каналу» LAP B (Link Access Procedure); практически совпадающая с канальным протоколом HDLC, а на сетевом уровне — протокол X.25/3. Действие этих трех протоколов носит локальный характер — в совокупности они организуют интерфейс между абонентской системой (ЭВМ, терминал) и сетью передачи данных, или, говоря точнее (с использованием связной терминологии), интерфейс между конечным оборудованием данных ООД (DTE) и конечным оборудованием канала данных ООК (DCE). Передача данных между АПД источника и АПД получателя производится согласно внут-

ренному протоколу сети, не оговариваемого X.25. Взаимодействие нелокального характера организует транспортный (при участии сеансового) уровень управления.

*Стандартный интерфейс X.21* определяет для случая дискретного канала связи с синхронной передачей физические, электрические и процедурные характеристики установления, поддержания и разъединения физического канала в точке между ООД и ООК, в том числе линии обмена (рис. 16.11), сигналы в интерфейсе и способ синхронизации символов между ЭВМ (терминалом) и каналом связи. Цепь синхронизации может отсутствовать. Тогда передаваемой через интерфейс последовательности управляющих сигналов предшествуют два или более символов SYN. При использовании цепи синхронизации по ней передаются 8-битовые синхронизирующие последовательности.

Более универсальным является протокол X.21 бис, позволяющий подключать абонентские и коммутационные системы как к дискретным, так и к аналоговым телефонным каналам.

*Протокол управления информационным каналом* — «Высокоуровневое управление каналов данных» (HDLC) <sup>1</sup> ориентирован на передачу последовательности бит и определяет следующие процедуры: формирование информационных и управляющих кадров, установление и прекращение связи, управление дуплексной передачей кадров, передача кадров и подтверждение об их приеме, формирование контрольных кодов и проверка правильности передачи, организация повторной передачи ошибочных кадров и др.

Канальный протокол HDLC предписывает формат кадра, представленный на рис. 16.12. Используются три типа кадров: информационный и два управляющих — супервизорный и нумерованный. Начало и конец кадра обозначаются уникальным кодом («флагом») 01111110 <sup>2</sup>.

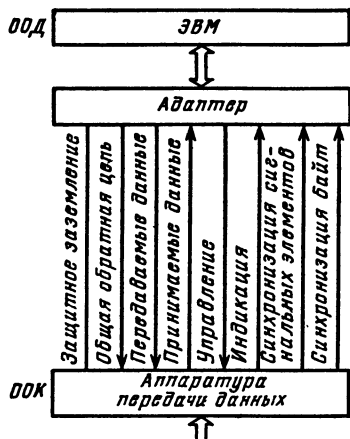


Рис. 16.11. Линии обмена ООД — ООК (протокол X.21)

<sup>1</sup> Высокоуровневое по отношению к физическому уровню управления.

<sup>2</sup> Прозрачность протокола по отношению к любой битовой последовательности обеспечивается с помощью процедуры «бит-стаффинг» (см. § 16.5).

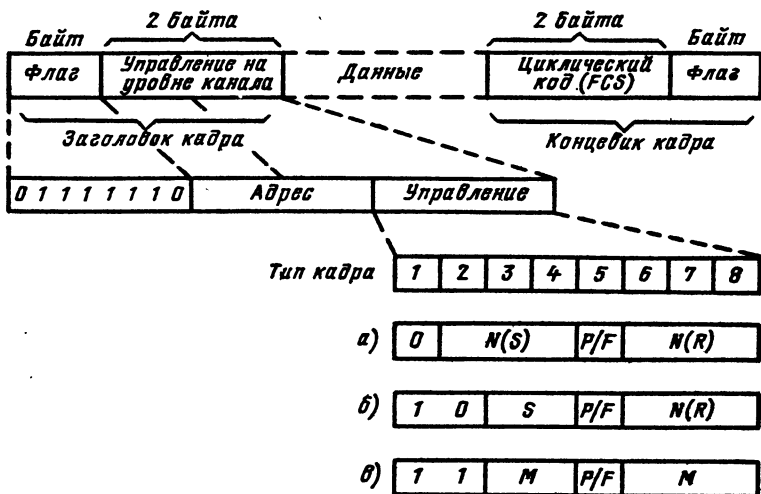


Рис. 16.12. Формат кадра канального протокола HDLC:  
 а — информационный; б — супервизорный; в — нумерованный

В целях контроля правильности передачи на передающей стороне формируется контрольный циклический код кадра (занимает в кадре 2 байта). На приемной стороне формируется снова контрольный код кадра и сравнивается с переданным контрольным кодом. В зависимости от результата сравнения формируется или не формируется подтверждение получения кадра.

В протоколе HDLC используется нумерация кадров по модулю 8 или по модулю 128. В последнем случае («расширенный режим») поля N(S) и N(R) увеличены до 7 бит и соответственно поле управления в кадре увеличено до 2 байт. В поле N(S) указывается порядковый номер (по соответствующему модулю) кадра, а в поле N(R) — номер следующего ожидаемого кадра для потока данных противоположного направления.

На передающей и принимающей сторонах устанавливаются счетчики N(S) и N(R). Эти счетчики и соответствующие поля в кадре используются для организации «окна», устанавливающего допустимое количество передаваемых в сеть кадров без получения подтверждения о получении.

Протокол HDLC предусматривает для каждой двух систем (абонентских, коммутационных), соединенных информационным каналом, присвоение одной системе статуса *первичная* (управляющая), а другой *вторичная* (управляемая). Первичная система управляет обменом информацией, посылая во вторичную

Т а б л и ц а 16.1. Состав кадров канального протокола

Название кадра	Мнемоническое обозначение	Функция
Информационный	I	C/R
Супервизорные:		
готовность к приему	RR	C/R
неготовность к приему	RNR	C/R
отказ	REJ	C/R
селективный отказ	SREJ	C/R
Ненумерованные:		
установить режим нормальных ответов (расширенный)	SNRM(E)	C
установить режим асинхронных ответов (расширенный)	SARM(E)	C
установить сбалансированный асинхронный режим (расширенный)	SABM(E)	C
прекратить связь	DISC	C
установить режим инициализации	SIM	C
запрос режима инициализации	RIM	R
запрос передачи (ненумерованный)	UP	C
сброс	RSET	C
сброс ненумерованный информационный	UI	C/R
обмен идентификаторами	XID	C/R
ненумерованное подтверждение	UA	R
режим разъединения	DM	R
запрос разъединения	RD	R
отказ от кадра	FRMR	C/R

П р и м е ч а н и е. C — команда; R — ответ; C/R — может использоваться как команда или ответ.

управляющие кадры, называемые *командами*, и получая от последней соответствующие управляющие кадры, называемые *ответами*.

Передача информационных кадров из первичной системы производится с использованием команд выбора (селекции) вторичной системы. Передача информационных кадров из вторичной системы организуется с помощью команд опроса, посылаемых первичной системой.

Для вторичной системы протокол HDLC предусматривает два режима работы: нормальный, при котором вторичная система может передавать кадры, только получив разрешение от первичной, и асинхронный, в котором вторичная система может производить передачу без разрешения первичной.

В табл. 16.1 приведены названия и мнемонические обозначения кадров канального протокола [13].

На рис. 16.13 представлены процедуры управления информационным каналом согласно канальному протоколу HDLC [69].

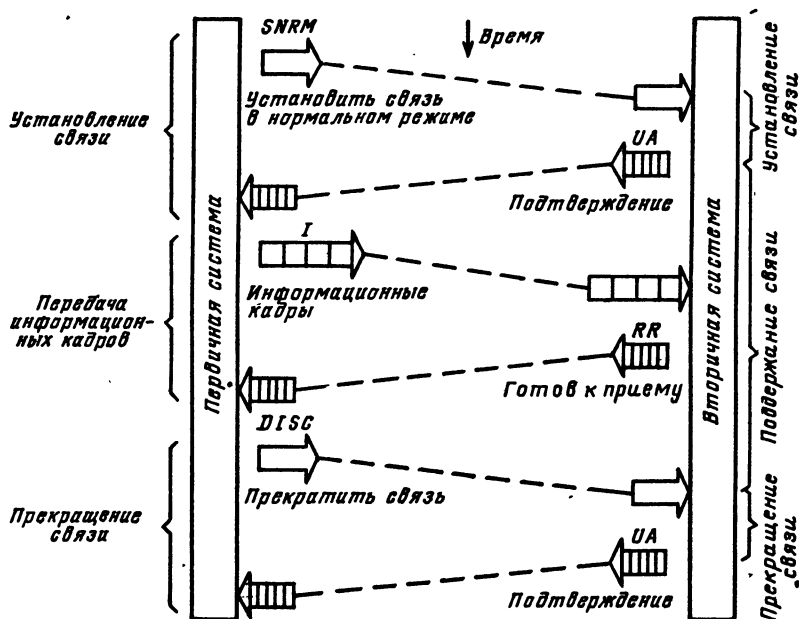


Рис. 16.13. Процедуры управления информационным каналом в канальном протоколе HDLC

*Сетевой протокол X.25/3.* Вычислительные сети с протоколом X.25 в принципе работают в режиме виртуальных каналов. Однако в заголовке пакета может быть установлен специальный индикатор, указывающий на то, что данный пакет является одиночным (дейтаграммой), содержит все подлежащие передаче данные, и поэтому для него нет необходимости устанавливать виртуальный канал между отправителем и адресатом.

Для простоты изложения материала далее предполагается, что абоненты (процессы пользователей) находятся в ГВМ, хотя все, что будет сказано, в равной степени относится к случаю, когда в качестве окончательного оборудования рассматривается терминал.

Пусть в ходе выполнения процесса пользователя в ГВМ понадобилось обеспечить его взаимодействие через сеть передачи данных с удаленным процессом в другой ГВМ, например передать массив данных. Для этого процесс регистрируется в транспортной станции своей ГВМ и становится абонентом сети. После этого между соответствующими процессами должно быть уста-

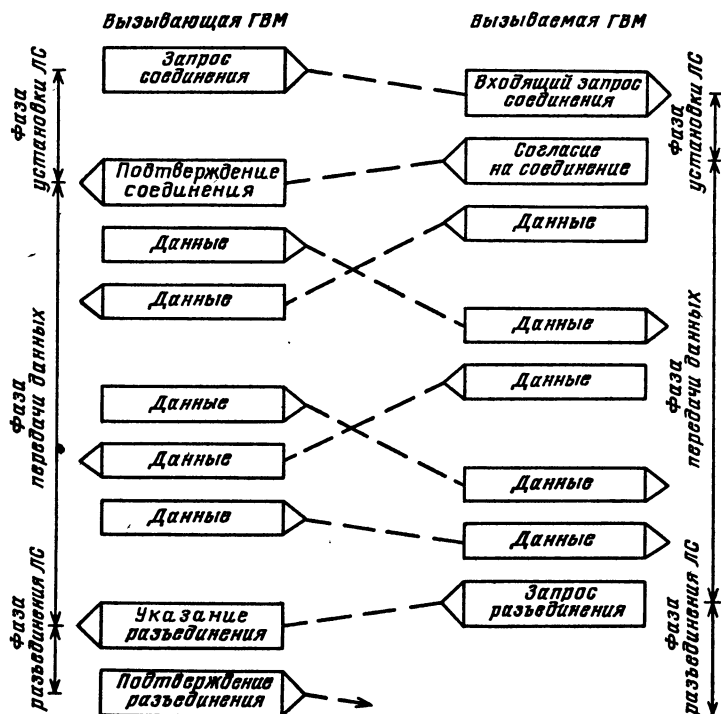


Рис. 16.14. Фазы взаимодействия ГВМ в соответствии с протоколом X.25/3

новлено логическое соединение (ЛС) — виртуальный канал для передачи данных.

Протокол X.25/3 в зависимости от условий взаимодействия абонентов регулирует создание временных ЛС на один сеанс связи и постоянных ЛС, которые нет необходимости устанавливать заново и разрывать при каждом сеансе связи. Главные вычислительные машины могут одновременно поддерживать несколько временных и (или) постоянных ЛС.

На рис. 16.14 показаны три фазы взаимодействия ГВМ: установки ЛС, передачи данных, разъединения ЛС.

Фаза установки временного ЛС протекает следующим образом. Запрашивающая соединение ГВМ передает в сеть передачи данных управляющий пакет *запрос соединения*, содержащий адреса обеих ГВМ, процессы в которых требуется соединить. Кроме того, в пакете могут указываться условия обслуживания: необходимые дополнительные средства (программы, устройст-

ва) и до 16 байт данных абонента-отправителя. Пакет *запрос соединения* на выходе из сети передачи данных преобразуется в пакет *входящий запрос соединения* для того, чтобы различать в одном физическом канале ГВМ-СП управляющие пакеты, направленные в разные стороны.

Если ГВМ-адресат согласен на установку ЛС, он передает в сеть связи пакет *согласие на соединение*, который на выходе из сети передачи данных преобразуется в пакет *подтверждение соединения*. После этого наступает фаза передачи данных.

Фаза разъединения ЛС может иметь место как после фазы передачи данных, так и во время фазы установки ЛС, когда вызываемая ГВМ не соглашается на установку ЛС и вместо пакета *согласие на соединение* направляет в сеть передачи данных пакет *запрос разъединения*.

После окончания сеанса связи, или при нарушении порядка передачи пакетов между ГВМ, или при перегрузке сети передачи данных любая из ГВМ может начать фазу разъединения передачей пакета *запрос разъединения*. На выходе из сети связи этот пакет преобразуется в *указание разъединения*. Запрашиваемая ГВМ должна ответить пакетом *подтверждение разъединения*, получение которого переводит ГВМ в состояние *готовность*: сетевая служба готова создать новое временное ЛС.

Пакеты с данными могут передаваться в любое время через постоянное ЛС или после установки временного ЛС. Абонент в ГВМ-отправителе выдает сообщение с данными для передачи удаленному абоненту в виде отдельных блоков. Программа управления передачей, относящаяся к четвертому (транспортному) уровню логической модели сети, добавляет к блоку заголовок, преобразуя его в фрагмент, и передает программе, реализующей протокол X.25/3, которая, в свою очередь, формирует пакет, добавляя к фрагменту заголовок пакета (заголовок X.25/3).

Полученный в результате пакет может быть одиночным (пакет дейтаграмма) или частью многопакетного сообщения (пакет данные). Хотя протокол X.25/3 устанавливает рекомендуемый максимальный размер поля данных в пакете, равный 128 байт, он разрешает администрации сети вводить и другие максимальные размеры в пределах 16—1024 байт, являющиеся степенью числа 2. На рис. 16.15 представлены форматы пакетов *данные* и *запрос соединения*.

Пакеты *данные*, *запрос соединения* и *дейтаграмма*, передаваемые или принимаемые в ГВМ по некоторому ЛС, циклически нумеруются от 0 до 7 (по модулю 8) или от 0 до 127 (по модулю 128). В первом случае заголовок пакета занимает 3 байта, во втором — 4 байта.

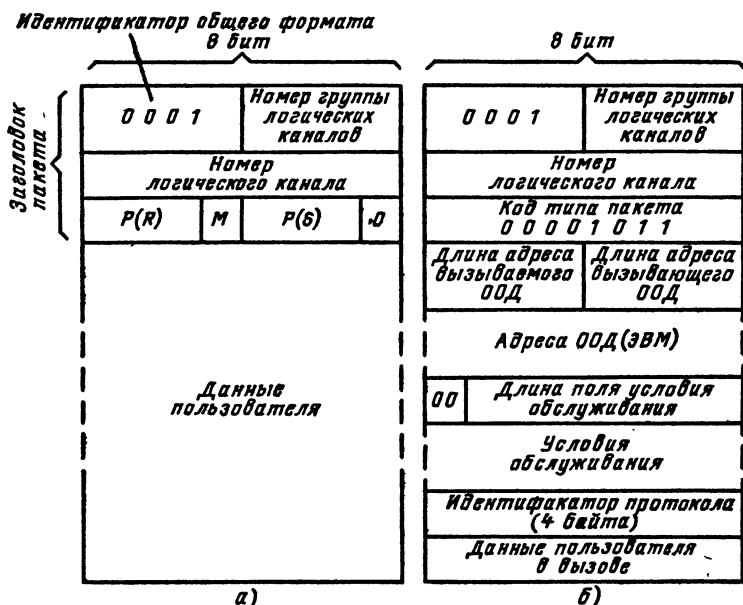


Рис. 16.15. Форматы пакетов данные (а) и запрос соединения (б) в протоколе X.25/3

В заголовке пакета помещаются два номера: присвоенный данному пакету порядковый номер передачи  $P(S)$  и порядковый номер  $P(R)$  пакета, последовательно принятого данной ГВМ (номер приема пакета). Порядковый номер передачи необходим для обеспечения последовательной транспортировки пакетов данных по виртуальному каналу, обнаружения потерь пакетов и управления интенсивностью поступления пакетов в сеть передачи данных.

Номер приема пакета используется как квитанция, подтверждающая прием пакетов адресатом и разрешающая передачу последующих пакетов в сеть передачи данных.

Если имеется поток пакетов только в одном направлении, то разрешения на передачу последующих пакетов передаются в виде специальных управляющих пакетов *готов к приему*. Если ГВМ (или СП) временно не может принимать пакеты с данными по некоторому ЛС, то они передают управляющий пакет *не готов к приему*.

В заголовке пакета *данные* имеется также специальный разряд  $M$ , принимающий значение 1, когда пакет является промежуточным. Если значение этого разряда равно 0, то пакет является последним в данной последовательности.



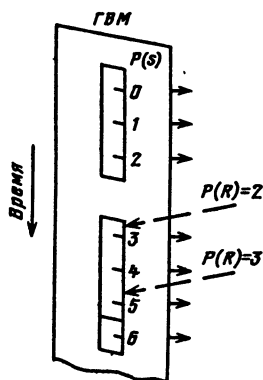


Рис. 16.16. Передача с использованием окна ( $w=3$ )

Размер окна с нумерацией пакетов по модулю 128 используется в сетях, в которых задержки при транспортировании пакетов велики. По мере увеличения загрузки сети передачи данных следует уменьшать размер окна, чтобы избежать переполнения буферов в узлах коммутации.

Следует отметить подобие структур процедур взаимодействия передающей и принимающей сторон на канальном (протокол HDLC) и сетевом (протокол X.25/3) уровнях (см. рис. 16.13 и 16.14).

После того как программа, реализующая протокол X.25/3, сформировала пакет, он передается программе, реализующей протокол канального уровня. В соответствии с этим протоколом пакет упаковывается в кадр: к пакету добавляют заголовок и концевик кадра. Затем этот кадр, согласно протоколу X.21 физического уровня и протоколу канального уровня, передается по каналу связи в следующий узел сети, где пакет из кадра передается программе, реализующей протокол X.25/3, которая считывает адрес назначения пакета и определяет канал для его дальнейшей передачи.

Когда пакет достигает ГВМ-адресата, из него извлекается фрагмент, который передается программе управления передачей, и данные поступают к абоненту-адресату.

## 16.7. Локальные вычислительные сети. Основные понятия

Локальной вычислительной сетью (ЛВС) называют сеть, все элементы которой — вычислительные машины (как правило, малые или микроЭВМ), персональные компьютеры, терминалы,

Для управления потоком пакетов в протоколе X.25/3 использован так называемый механизм окна. Согласно этому механизму фиксируется максимальное число  $w$  последовательно пронумерованных пакетов, которые разрешается передать без получения подтверждения об их приеме от адресата. Величина  $w$  называется размером окна. При нумерации пакетов по модулю 8 размер окна может достигать до 7, а при нумерации по модулю 128 — до 127.

Границы окна изменяются (окно продвигается) при получении номера приема  $P(R)$ , большего (по модулю 8 или 128), чем последний принятый номер  $P(R)$  (рис. 16.16). Окно большего

связная аппаратура — располагаются на сравнительно небольшой территории, например, в радиусе от нескольких сотен метров до 3—10 км.

Такая локальная сеть обычно предназначена для сбора, передачи, рассредоточенной и распределенной обработки информации в пределах одного предприятия или организации, часто специализируется на выполнении определенных функций в соответствии с профилем деятельности предприятия и отдельных его подразделений. Во многих случаях ЛВСт, обслуживая АСУ организационного типа на предприятии, связана, с одной стороны, с автоматизированными системами управления отдельными технологическими процессами и установками, а с другой стороны, с крупной региональной вычислительной сетью.

Наличие локальной сети позволяет специализировать и приближать обработку информации к местам, где она зарождается, а результаты обработки — к лабораториям, КБ, цехам, отделам и т. д., освобождая от ряда работ вычислительный центр организации и обеспечивая при этом быстрый обмен информацией между отдельными подразделениями и между подразделениями и главным ВЦ.

Эффективность локальной сети основана на том, что, как показывает практика, подавляющая часть информации, с которой имеют дело на предприятии (в организации), циркулирует между отдельными его (ее) службами или даже внутри подразделений и, таким образом, охватывается локальной сетью.

Значение создания локальных сетей в последнее время сильно возросло благодаря широкому масштабу распространения персональных компьютеров, созданию на их основе различных автоматизированных рабочих мест (АРМ), для эффективного применения которых на предприятии или в организации во многих случаях необходимо иметь возможность обмена информацией между отдельными АРМ (работниками предприятия) и между АРМ и главным ВЦ.

Наличие локальной сети позволяет упростить и удешевить оборудование рабочих мест, в том числе персональных компьютеров вследствие коллективного использования ими через локальную сеть (в режиме разделения времени) наиболее дорогих ресурсов, таких, как дисковая память большой емкости, быстродействующие и высококачественные печатающие устройства.

На рис. 16.17 представлены типичные структуры ЛВСт с шинной и кольцевой топологиями. Реже встречаются звездообразные и древовидные ЛВСт.

Абонентами ЛВСт могут быть ЭВМ разного типа, *рабочие станции* в виде персональных компьютеров (АРМ) и терминалов, за которыми работают пользователи сети, различные сервисные

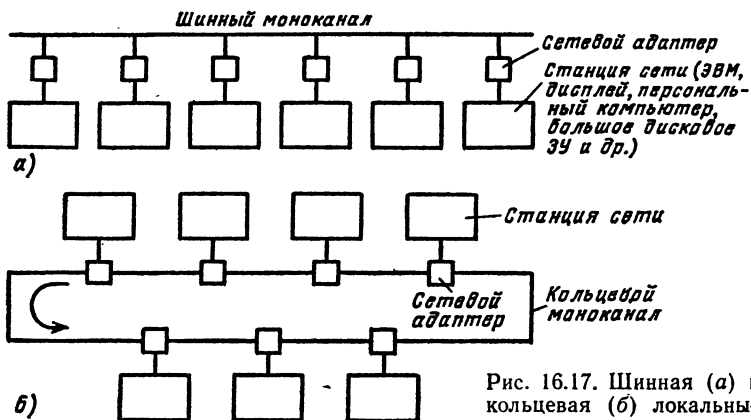


Рис. 16.17. Шинная (а) и кольцевая (б) локальные вычислительные сети

станции, в том числе *файловый сервер* на основе дискового ЗУ большой емкости, *связной сервер*, *сервер печати*. Абоненты сети подключаются к шине (магистральной) или кольцу при помощи *сетевых адаптеров* (СА), называемых также *сетевыми контроллерами*. Сетевые адаптеры через свои приемопередатчики (ПП) производят прием сигналов из канала, выдачу их в канал, преобразование форматов данных для согласования реализованной в локальной сети последовательной передачи (последовательного интерфейса) с внутренним параллельным интерфейсом устройств (станций), подключенных к сети, а также в зависимости от особенностей ЛВСт выполняют некоторые другие функции.

Снабженное адресом сообщение (кадр) от узла-источника в шинной ЛВСт практически одновременно поступает ко всем узлам сети, а в кольцевой последовательно ретранслируется (с небольшой задержкой) от узла к узлу сети (эстафетная передача), пока не достигнет (в качестве квитанции) узла-источника, снимающего сообщение с кольца. В обоих случаях только узел, распознавший в сообщении свой адрес, принимает сообщение в регистры своей станции.

Таким образом, вся коммутационная сеть передачи данных ЛВСт сжимается до одного коммутационного узла, в роли которого выступает один общий канал (в виде шины или кольца). Этот общий канал получил название *моноканала*.

Наличие моноканала является принципиальной особенностью ЛВСт, определенным образом модифицирующей эталонную модель вычислительной сети (рис. 16.18). В ЛВСт благодаря наличию моноканала отпадает необходимость в маршрутизации сообщений, а поэтому в модели исчезает сетевой уровень

управления, но одновременно усложняется канальный уровень, так как появляется необходимость в селекции сообщений — проверке каждым узлом, ему ли сообщение адресовано. Кроме того, становится более сложной процедура доступа к моноканалу по сравнению с процедурой доступа в обычной сети передачи данных. Вследствие этого в ЛВСст целесообразно разделение канального уровня управления (а следовательно, и канального протокола) на два подуровня: 2.1. *Управление доступом к передающей среде* (MAC — Medium Access Control) и 2.2. *Управление логическим каналом* (LLC — Logical Link Control), которое выполняет функции селекции (адресации), контроля ошибок и управления протоколом данных.

Протоколы уровней управления 1, 2, а часто и 4 (уровень 3 в ЛВСст исчезает) реализуются в ЛВСст аппаратурой сетевого адаптера, а более высоких уровней — программными средствами станций сети.

Различают *однородные ЛВСст* с программно-совместимыми абонентами (например, с персональными компьютерами одного типа) и *неоднородные*, объединяющие, например, различные персональные компьютеры, разнообразные программируемые устройства технологического оборудования.

Структура локальной сети отображает (в определенных пределах) структуру обслуживаемой организации, а поэтому часто имеет иерархическое построение. При этом ЛВСст, обслуживающие отдельные подразделения и формы работ (автоматизация проектирования, испытаний, технологических процессов, информационное обслуживание, планово-экономическое и оперативное управление), образуют многоуровневую систему взаимодействующих сетей (*интерсеть*). Связь между сетями обеспечивается аппаратурно-программными средствами специальных узлов интерсети, называемых *мостами*, если они связывают однотипные (с одинаковыми протоколами) сети, и *шлюзами*, если они связывают разнотипные сети, требующие для взаимодействия выполнения процедур преобразований (преобразование форматов данных и т. п.). Примеры интерсетей приведены на рис. 16.24—16.26.

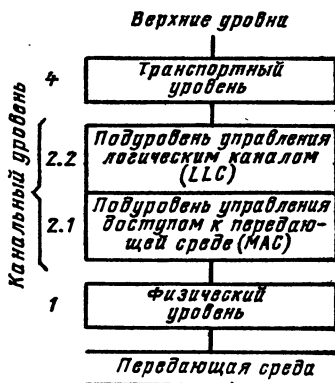


Рис. 16.18. Модификация эталонной модели вычислительной сети применительно к локальной сети

В интересах надо осуществлять маршрутизацию сообщений при передаче их из одной локальной сети в другую, что приводит к необходимости реализации сетевого уровня управления.

Распространенной на практике является двухуровневая локальная интересь, нижний уровень которой составляют одна ЛВСст или несколько сетей, объединяющих (в целях автоматизации производственных процессов) существенно неоднородное программно-управляемое цеховое производственное оборудование, а верхний уровень представлен ЛВСст, обслуживающей конструкторскую и технологическую подготовку производства и его планово-экономическое управление. Такова, например, интересь ТОР/МАР, рассмотренная в § 16.9.

### **16.8. Особенности организации передачи информации в локальных сетях. Методы доступа к моноканалу**

При построении ЛВСст важно найти сравнительно простые и дешевые решения для реализации связи между объединяемой сетью устройствами. В ЛВСст используются элементы связной технологии; в частности, передача информации по моноканалу производится последовательным кодом.

Используемая в ЛВСст передающая среда, скорость и способ передачи информации определяются назначением сети, ее топологией, уровнем помех, создаваемых производственным оборудованием, окружающим компоненты сети.

Тип передающей среды в первую очередь зависит от требований к скорости передачи информации в сети. При скорости передачи от нескольких сотен килобит в секунду до 1 Мбит/с используют витые (экранированные) пары проводов, при скорости от 1 до 20 Мбит/с — помехозащищенный коаксиальный или полностью не подверженный действию помех оптоволоконный кабель.

Благодаря небольшим расстояниям между компонентами ЛВСст, ограничивающим влияние на ее работу помех, широкое распространение в локальных сетях получила прямая передача дискретной информации, при которой цифровые сигналы (0,1) прямо, без модуляции несущей частоты, поступают в моноканал.

К ЛВСст, используемым для построения систем автоматизации производственных процессов, предъявляются сравнительно умеренные требования в отношении скорости передачи данных (обычно 200—500 Кбит/с). Однако в ряде случаев в таких сетях из-за высокого уровня промышленных помех приходится отказываться от прямой передачи дискретной информации и применять передачу данных с модуляцией несущей частоты, обладающую большей помехоустойчивостью.

В ЛВСт учреждений, НИИ, конструкторских бюро, в которых должны реализовываться передачи файлов, изображений, режим «электронной почты», требования к скорости передачи возрастают до нескольких мегабит в секунду, но из-за низкого уровня помех сохраняется возможность прямой передачи дискретной информации. В ближайшем будущем станет актуальной передача по локальным сетям наряду с данными также речевой и телевизионной информации, что потребует реализации в этих сетях широкополосной передачи, при которой для разных видов сообщений в моноканале выделяются определенные полосы (определенные несущие частоты).

**Синхронизация передачи.** В ЛВСт при передаче сообщений должна обеспечиваться синхронизация работы приемника и передатчика, с тем чтобы приемник распознавал интервалы времени (такты) представления бит в передаваемом сообщении. В ряде сетей, например в сети Cambridge Ring, это достигается с помощью дополнительной линии, по которой передаются сигналы отметок времени (бит). В сетях с передачей сообщений в основной полосе частот, т. е. при передаче немодулированных сигналов, синхронизация обеспечивается без дополнительных линий путем применения самосинхронизирующегося манчестерского или дифференциального манчестерского кода (рис. 16.19), в котором посередине каждого интервала времени представления бита (такта) происходит смена уровня сигнала: для манчестерского кода с нижнего уровня на верхний при передаче 1 и с верхнего на нижний при передаче 0; для дифференциального манчестерского кода при 0 сохраняется направление перепада предыдущего бита, а при 1 оно меняется на обратное.

Типичный формат пакета для локальных сетей, приведенный на рис. 16.20, содержит адреса получателя и отправителя, так

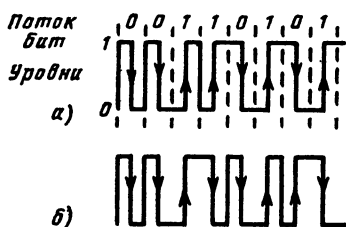


Рис. 16.19. Манчестерское (а) и дифференциальное манчестерское (б) кодирование

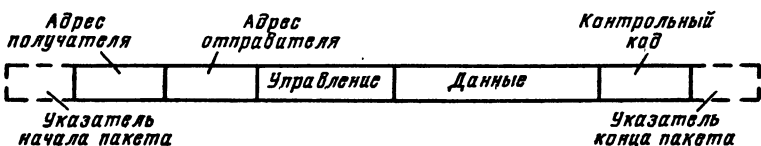


Рис. 16.20. Типичный формат пакета для локальных сетей

как в отличие от сетей с узлами коммутации, где, как правило, используются двухточечные соединения, получатель всегда знает, откуда поступил пакет (кадр), и адрес отправителя можно в кадре не указывать, в ЛВСт с моноканалом факт поступления сообщения ничего не говорит об отправителе.

*Методы доступа* регламентируют процедуры получения абонентами доступа к моноканалу. Необходимость такой регламентации вызывается тем, что при одновременной передаче данных несколькими абонентами возникает конфликтная ситуация — наложение и взаимное искажение информации.

Управление доступом станции к моноканалу может быть централизованным и осуществляться специальной управляющей (мониторной) станцией либо децентрализованным — распределенным между всеми станциями сети. В ЛВСт используются детерминированные и случайные методы доступа к моноканалу, причем последние применяются главным образом в шинных ЛВСт.

Наличие централизованного управления снижает надежность ЛВСт, так как при выходе из строя только одной мониторинговой станции сеть становится неработоспособной. В сетях со случайным доступом из-за возможности конфликтов не гарантируется максимальная задержка передачи сообщения. Поэтому сети со случайным доступом могут оказаться непригодными для использования в системах, осуществляющих управление в реальном масштабе времени.

В кольцевых ЛВСт, реализующих эстафетную передачу данных, применяются в основном три метода доступа к моноканалу: метод вставки регистра, метод временных сегментов или тактированного доступа, метод маркерного доступа (передачи маркера).

*Метод вставки регистра* (рис. 16. 21, а). Оборудование каждого узла кольцевой сети содержит буферные сдвигающие регистры  $R_{гБ1}$ ,  $R_{гБ2}$  одинаковой емкости и электронный переключатель, который в положении 1 соединяет входную и выходную линии узла (отключает станцию от выходной линии), а в положениях 2 и 3 подключает к выходной линии выходы сдвигающих регистров соответственно  $R_{гБ2}$  и  $R_{гБ1}$ .

Вход сдвигающего регистра  $R_{гБ1}$  всегда соединен с входной линией, и в него поочередно поступают передаваемые по кольцу сообщения. Если поступившее в  $R_{гБ1}$  сообщение распознается как адресованное данному узлу, оно поступает на станцию узла, при этом сообщение продолжает передаваться по кольцу до тех пор, пока не достигнет станции-отправителя (в качестве квитанции о получении сообщения).

Когда станция готова передать сообщение, она загружает

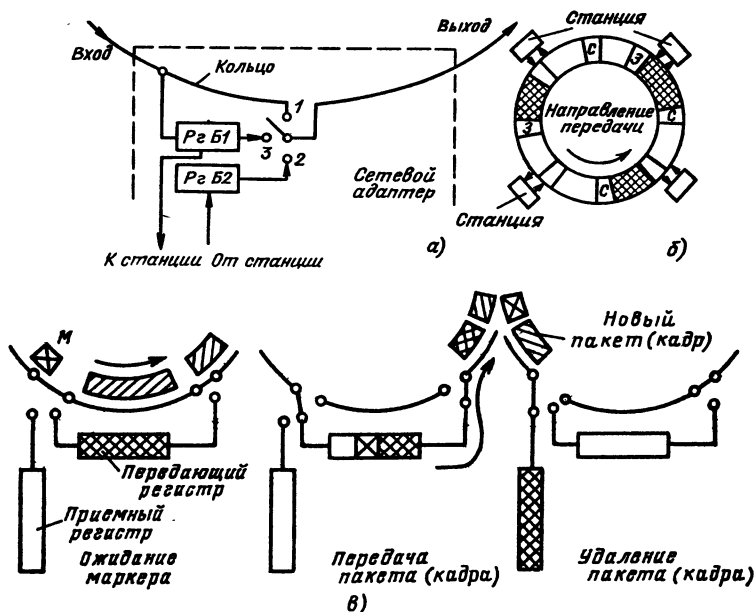


Рис. 16.21. Методы доступа в кольцевой локальной вычислительной сети:  
а — метод вставки регистров; б — метод временных сегментов; в — маркерный доступ

его в сдвигающий регистр  $P_2B2$ . Передаче должно предшествовать состояние узла, соответствующее положению 1 переключателя. В этом состоянии станция (сетевой адаптер) проверяет, происходит ли в данный момент через ее узел ретрансляция сообщения, и ждет его окончания. Затем переключатель устанавливается на определенное время в положение 2, и подготовленное сообщение из  $P_2B2$  поступает в кольцевую сеть. Если во время передачи сообщения на вход узла поступит другое сообщение, оно будет принято на буферный сдвигающий регистр  $P_2B1$ .

После окончания передачи своего сообщения станция устанавливает переключатель в положение 3, и в кольцевую сеть передается сообщение, сохраненное в буферном регистре  $P_2B1$ . Переключатель остается в положении 3 до того момента, когда в  $P_2B1$  поступит с кольца посланное станцией сообщение, что служит квитанцией о его передаче. Переключатель снова устанавливается в положение 1.



Рассматриваемый вариант предполагает централизованное управление, при котором мониторинговая станция обнаруживает и удаляет сообщения с отсутствующими в сети адресами, осуществляет синхронизацию работы кольцевой сети. Возможна одновременная передача сообщений несколькими станциями. Длина сообщений не фиксирована, но ограничена емкостью буферных регистров.

К недостаткам кольцевых сетей с вставкой регистра можно отнести изменяющуюся из-за вставки регистров длину кольца и высокие требования, предъявляемые к быстродействию электронного переключателя регистров.

*Метод временных сегментов* (тактированного доступа)<sup>1</sup> (рис. 16.21, б). В кольцевом канале связи сигналами мониторинговой станции выделяются временные сегменты фиксированной длины.

Сегменты циркулируют по кольцу, и каждая станция может поместить свой пакет данных (кадр) или его часть в один из них, если он помечен как «свободный» (С), при этом в сегменте метка «свободный» заменяется меткой «занятый» (З). После доставки пакета адресату сегмент снова освобождается.

Данный вариант построения кольцевой ЛВС требует централизованного управления, допускает одновременную передачу данных несколькими станциями. Передача данных производится пакетами (кадрами) фиксированной длины.

*Метод передачи маркера* (рис. 16.21, в). По свободному от передачи сообщений кольцу от одного сетевого адаптера (ПП) к другому циркулирует маркер М (определенная последовательность символов), регенерируемый мониторинговой станцией.

Передачу сообщений может производить только та станция, которая получила маркер. Как только на подготовившую сообщение станцию (точнее, ее сетевой адаптер) поступит маркер, его дальнейшее перемещение по кольцу временно блокируется. Начинается передача подготовленного сообщения, которая завершается передачей маркера, при этом ПП каждой станции с небольшой задержкой, равной, например, половине интервала представления бита, ретранслирует сообщение соседней станции.

Соседняя станция транслирует все биты и далее может передать либо собственное сообщение (завершив передачу маркером), либо маркер. В соответствии с адресом, указанным в начале сообщения, последнее будет воспринято соответствующей станцией как ей адресованное. Станция-адресат в служебном поле сообщения фиксирует факт его приема и результат контроля правильности принятого сообщения.

---

<sup>1</sup> Этот метод доступа применен в английской кольцевой сети Cambridge Ring.

Сообщение продолжает двигаться по кольцу до станции-отправителя. Эта станция воспринимает свое сообщение как квитанцию и в случае отсутствия ошибок удаляет его из кольца; в противном случае она организует повторную передачу.

Кольцевая ЛВС с циркулирующим маркером позволяет передавать сообщения произвольной длины, но не допускает одновременную передачу сообщений несколькими станциями, требует централизованного управления.

Характерными нарушениями работы ЛВС с маркерным доступом являются пропадание маркера и появление лишних маркеров. Мониторная станция, используя тайм-аут, контролирует пропадание и размножение маркера: регенерирует пропавший и снимает с кольца лишний маркер.

К общему недостатку кольцевых сетей следует отнести их невысокую живучесть: достаточно выйти из строя одной станции или линии связи на одном участке кольца и сеть становится неработоспособной. Поэтому при выходе из строя сетевого адаптера (или станции) предусматривается возможность продолжения работы его приемника-передатчика в качестве простого ретранслирующего устройства. Были предложены конфигурации, использующие несколько петель. Такие системы менее критичны к отдельным неисправностям. При возникновении отказов может быть произведена реконфигурация с целью восстановить хотя бы одно кольцо. Наличие дополнительных линий связи позволяет повысить пропускную способность системы.

В шинных ЛВС широко используются случайные методы доступа, причем наибольшее распространение получил *Множественный доступ с контролем несущей и обнаружением конфликтов (CSMA/CD)*<sup>1</sup> [14,62].

Станция, подготовившая сообщение для передачи, прослушивает моноканал (контроль несущей) и ждет его освобождения от передачи сообщений от других станций. Как только моноканал освобождается, станция начинает передачу своего сообщения, но при этом продолжает прослушивать моноканал и сравнивать передаваемые ею сообщения с сообщениями в моноканале. Их несовпадение (искажение передаваемых сообщений) указывает на конфликтную ситуацию, вызванную одновременной передачей сообщений в моноканале другой станцией. Обнаружившая конфликт станция прекращает передачу и повторяет попытку занять моноканал через некоторое случайное время, среднее значение которого увеличивается (например, вдвое) с каждой неудачной попыткой произвести передачу.

---

<sup>1</sup> CSMA/CD — Carrier-sense Multiple Access with Collision Detection.

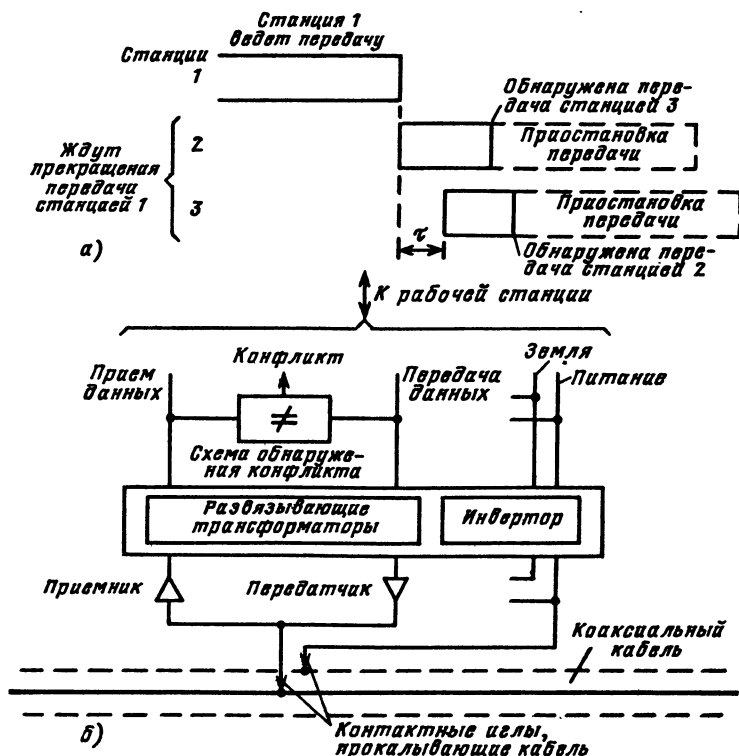


Рис. 16.22. Множественный доступ с контролем несущей и обнаружением конфликтов (CSMA/CD):  
а — процедура доступа; б — схема обнаружения конфликтов

На рис. 16.22 показаны процедура рассматриваемого случайного доступа к шинному моноканалу и схема, позволяющая обнаруживать конфликты. Существенную роль играет задержка  $\tau$  распространения сигнала по шинной сети (неодновременность поступления сигналов к различным станциям). Пусть станция 1 ведет передачу, а станции 2 и 3 подготовили сообщения и прослушивают моноканал. Из-за задержки в сети  $\tau$  станции 2 и 3 в разные моменты времени обнаруживают освобождение моноканала и начинают передачу своих сообщений. Со сдвигом по времени  $\tau$  они обнаружат конфликтную ситуацию и прекратят передачу. С увеличением протяженности сети увеличиваются  $\tau$  и вероятность конфликтов.

В шинных ЛВСт применяется также упрощенный вариант рассматриваемого случайного доступа, когда конфликт не обна-

Уровни

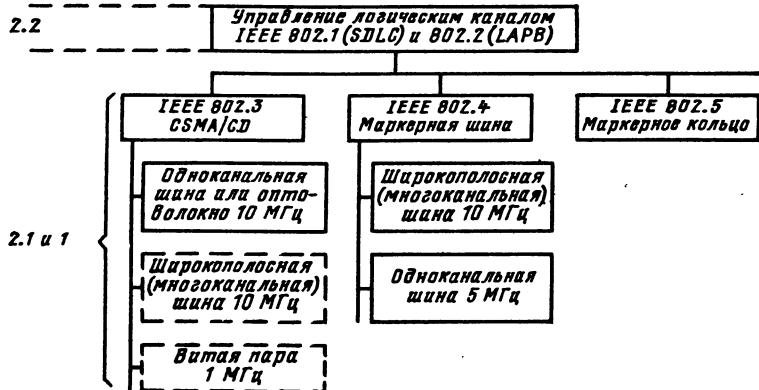


Рис. 16.23. Схема стандартизации протоколов локальных вычислительных сетей

руживается в ходе передачи и конфликтующие передачи продолжаются до конца, а конфликтная ситуация устанавливается по факту неправильной передачи сообщения. Этот вариант случайного доступа по сравнению с предыдущим имеет меньшую пропускную способность канала из-за больших потерь времени при конфликтах.

В шинных ЛВС применяется также маркерный доступ, при котором «маркерная шина» образует логическое кольцо, а также тактированный доступ.

**Стандартизация протоколов ЛВС.** Широкое распространение ЛВС, разнообразие решаемых с их помощью задач, разнотипность объединяемого локальной сетью оборудования, возникшая тенденция поставки станков и другого технологического оборудования со встроенными сетевыми средствами сделали остро актуальным международную стандартизацию локальных сетей. Активную деятельность в разработке проектов этих стандартов проявляет комитет 802 IEEE (США).

Международные стандарты обычно создаются на основе опробованных практикой решений фирм-изготовителей, получивших широкое распространение и признание, при этом центральное место в стандартизации ЛВС занимают процедуры обмена информацией.

На рис. 16.23 даны принятые или рассматриваемые (показаны штриховыми линиями) стандарты на протоколы канального и физического уровней локальных сетей [1]. На рисунке приведены обозначения протоколов, тип сети (шинная, кольцевая), используемая передающая среда, метод доступа и скорость

передачи данных. Протоколы более высокого уровня (3—7) ЛСт рассматриваемыми стандартами не регламентируются.

### **16.9. Примеры локальных сетей: сеть TOP/MAP, сеть Ethernet, сеть IBM с маркерным доступом**

Локальная сеть TOP/MAP<sup>1</sup> представляет собой двухуровневую интерсеть. Нижний уровень — сеть MAP (Manufacturing Automation Protocol), ориентированная на автоматизацию управления технологическим оборудованием (роботы, программные контроллеры, станки с числовым программным управлением и т. п.) в цеховых условиях. Второй уровень — сеть TOP (Technical and Office Protocol) объединяет АРМ работников административной, конструкторской и технологической служб предприятия.

Взаимодействие этих сетей реализуется на уровне обмена файлами.

Протоколы TOP/MAP фактически стали международным стандартом при построении сетей предприятий.

В сетях TOP/MAP реализуются все семь уровней эталонной модели открытых систем ИСО (см. § 16.4).

*Локальная сеть MAP* предназначена для построения интегрированных систем автоматизации производства, объединяющих с помощью этой сети разнообразные средства, управляющие различным производственным оборудованием. Важнейшая особенность сети MAP состоит в том, что она является неоднородной сетью, позволяющей единым образом организовать связь и взаимодействие разнотипного и несовместимого друг с другом оборудования — различных программируемых контроллеров, устройств УПУ и т. д.

Ориентация сети MAP на работу в цеховых условиях, где возможны сильные помехи от силового электрооборудования, предопределила передачу данных в сети с использованием несущей частоты, обладающую большей помехоустойчивостью по сравнению с помехоустойчивостью при прямой передаче дискретной информации. Необходимость обеспечить управление технологическим оборудованием в реальном масштабе времени потребовала высокой скорости передачи данных, обеспечения небольшой максимальной задержки в передаче сообщения.

Сказанное определило выбор для сети MAP детерминированного управления доступом к сети, построение сети на основе

---

<sup>1</sup> Сеть TOP/MAP объединяет взаимодействующие сети: MAP (разработка фирмы General Motors) и TOP (разработка фирмы Boeing).

шины с маркерным доступом, использование в качестве передающей среды коаксиального кабеля с широкополосной (многоканальной) и моноканальной передачей на несущих частотах со скоростью 10 или 5 Мбит/с.

На уровнях управления 1 и 2.1 сеть MAP соответствует стандартному протоколу IEEE 802.4, а на уровне 2.2 — протоколу 802.2 (аналогичному канальному протоколу LAR B).

На более высоких уровнях программным путем реализуются протоколы согласно рекомендациям ИСО, при этом представительный уровень, как таковой, в MAP отсутствует, его функции выполняются на прикладном уровне.

*Мини-MAP* — локальная сеть, используемая в небольших сетях для управления в реальном времени сравнительно небольшим числом гибких автоматизированных модулей, станков с ЧПУ, программируемых контроллеров. Реализуется быстрый прямой доступ с прикладного уровня управления к физическому каналу в соответствии с «Протоколом управления технологическим оборудованием» 9506. Сеть мини-MAP строится в виде соответствующей протоколу 802.4 шины с маркерным доступом (логическое кольцо), использующей в качестве моноканала коаксиальный кабель с передачей сообщений на несущей частоте (с модуляцией и демодуляцией) со скоростью 5 Мбит/с.

*ТОР* — *локальная сеть*, предназначенная для организации связи и взаимодействия АРМ работников, занятых автоматизированным проектированием и технологической подготовкой производства, планово-экономическим и оперативным управлением на предприятии.

Сеть работает в конторских помещениях, залах конструкторов и технологов, где низок уровень помех. Отсутствует требование реализации управления в реальном масштабе времени, но сохраняются требования к высокой скорости передачи. Поэтому сеть имеет более простую по сравнению с MAP архитектуру — шину со случайным доступом с контролем несущей и обнаружением конфликтов (CSMA/CD) с использованием коаксиального кабеля для одноканальной прямой передачи дискретных сообщений или широкополосной многоканальной передачи со скоростью 10 Мбит/с.

В сети *ТОР* используется стандартный протокол IEEE 802.3. Эта сеть близка к широко распространенной в мировой практике локальной сети Ethernet.

*Локальная сеть Ethernet*<sup>1</sup> предназначена в первую очередь

---

<sup>1</sup> Разработана фирмой Хегох, к которой затем присоединились фирмы DEC и Intel (США).

для объединения и взаимодействия электронного оборудования учреждений. При ее разработке преследовалась цель обеспечить простоту и дешевизну сетевых средств по сравнению с устройствами, которые присоединяются к сети (персональные компьютеры, учрежденческие АРМ и др.), высокую надежность, устойчивость при отказах ее отдельных устройств, низкие эксплуатационные расходы, достаточную пропускную способность при пульсирующем трафике, характерном для учрежденческих сетей, простоту расширения сети.

Выполнению этих требований способствуют децентрализованное управление сетью, случайный доступ к сети, шинная структура, допускающая при расширении сети объединение нескольких шин с образованием древовидной топологии (рис. 16.24).

Главным в сети Ethernet является впервые предложенный при ее разработке множественный доступ с контролем несущей и обнаружением конфликтов (CSMA/CD).

Передающей средой служит коаксиальный кабель с сопротивлением 50 Ом, причем максимальная длина сегмента кабеля 500 м. Каждый сегмент должен иметь на обоих концах согласующую нагрузку 50 Ом. Сегменты соединяются через повторители. Максимальное количество станций в сети 1024. Расстояние между точками присоединения станций к моноканалу должно быть не менее 2,5 м. Оптимальным является расположение точек присоединения станций на расстоянии от начала кабеля, кратном 2,5 м.

Наиболее простой способ передачи сигнала — прямая (немодулированная) последовательная передача дискретных сообщений с использованием манчестерского кодирования.

Скорость передачи данных 10 Мбит/с. Кадр имеет переменную длину: от 72 до 1526 бит. Передача кадра начинается с передачи кода — преамбулы (64-битная последовательность вида 1010... 10101011), настраивающей на прием другие станции сети.

*Звездно-кольцевая локальная вычислительная сеть*

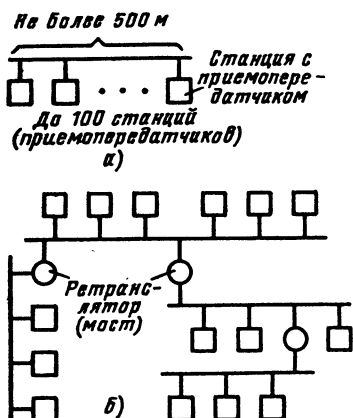


Рис. 16.24. Варианты конфигурации локальной сети Ethernet: а — простейшая; б — с использованием ретрансляторов (мостов)

с маркерным доступом фирмы IBM [73]. При разработке локальной сети учитывалось, что в ближайшем будущем потребуются сети с возможностью работы с различными данными: цифровыми, текстовой информацией, неcodируемыми данными (факсимильная информация), аналоговыми и цифровыми речевыми сообщениями, аналоговой видеoinформацией. При этом должны учитываться различия при работе с разными данными в требованиях к скорости передачи (полосе частот) и в отношении контроля ошибок (при передаче факсимильной, речевой или видеoinформации не требуется контроль ошибок). Разнообразие типов данных предопределяет использование передачи сигналов как немодулированных, так и модулированных. Принято, что данные в ЛВСт могут передаваться пакетами, состоящими из информационного поля переменной длины, адресных и служебных полей. Выбор топологии сети определялся компромиссом между стремлением минимизировать длину кабеля (в значительной степени определяющего стоимость сети) и упростить реконфигурацию сети при отказах ее компонентов.

Минимальная длина кабеля достигается при последовательном соединении узлов. Но при очень большом числе узлов (несколько сотен) высоки затраты на обслуживание и затруднен выбор конфигурации сети. При звездообразной топологии с одним пунктом концентрации нет указанных недостатков, но велика общая длина кабеля. Сказанное привело к смешанной, звездо-кольцевой топологии, в которой кольцо с маркерным доступом представляет собой самостоятельный архитектурный компонент (рис. 16.25), допускающий реализовывать сравнительно простые кольцевые ЛВСт, а при большем числе узлов

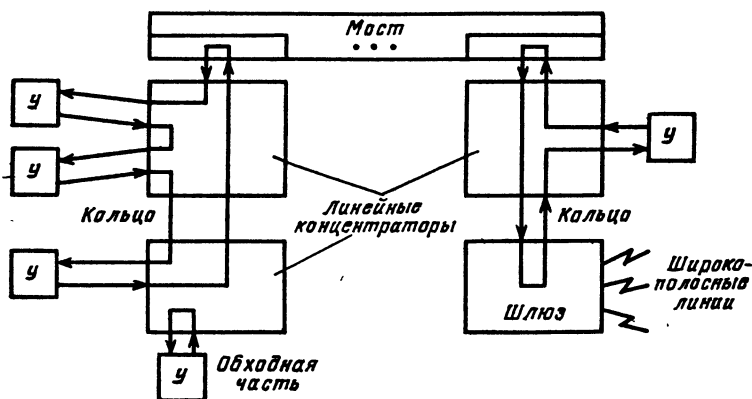


Рис. 16.25. Основные компоненты кольцевой ЛВСт фирмы IBM:  
У — узел сети



использовать звездную структуру для объединения нескольких кольцевых сетей.

Рассматриваемая кольцевая сеть с маркерным доступом послужила основой стандарта IEEE 802.5. Компонентами сети являются узлы (ЭВМ, персональные компьютеры, концентраторы линий, мосты и шлюзы). Узлы подключаются к сети через сетевые адаптеры. Передающей средой служат витые пары проводов с заземляемым экраном (при передаче со скоростью 4 Мбит/с) и оптоволоконные кабели (при скорости 10 Мбит/с и более). Рекомендуется применение оптоволоконных кабелей для передачи данных между зданиями, не требующими специальных мер по защите от молний. В ЛВСт могут использоваться имеющиеся в здании телефонные и кабельные телевизионные сети. Передача дискретных сообщений производится с помощью манчестерского кодирования.

Концентраторы линий, к которым подключаются узлы, придают сети гибкость (возможность построения различных вариантов структур, в том числе иерархических) и надежность (обеспечивают возможность реконфигураций), облегчают эксплуатацию сети.

Узлы подключаются к отдельным секциям концентраторов. Внутри концентраторы секции соединены последовательно. Сами концентраторы соединяются последовательно в кольцо. Концентраторы могут соединяться друг с другом сегментами оптоволоконного кабеля, а узлы подключаться к концентраторам, экранированным витым двухжильным кабелем. Мосты — высокоскоростные переключающие устройства — служат для связи между отдельными кольцевыми сетями. В одно кольцо не удастся объединить более 100—200 узлов из-за сложности управления и «дрожания» низкой частоты, вызываемого наличием большого числа ретрансляторов, что может нарушить синхронизацию. Если в системе более 100—200 узлов или эти узлы разбросаны по большей площади, рекомендуется построение нескольких колец и организация связи между ними через мосты. Мост по отношению к кольцу выступает как один из ее узлов. Мост может связывать кольца с разной скоростью передачи сообщений. В очень больших сетях (несколько сотен узлов) предусматривается иерархическая структура (интерсеть), в которой несколько колец соединяются с помощью высокоскоростных магистралей, выполняемых в виде кольца или шины (рис. 16.26). При наличии в системе мостов и шлюзов должна производиться маршрутизация пакетов, обеспечиваемая сетевым уровнем управления. Функционирование моста должно быть прозрачным для пользователя сети, т. е. для него вся сеть должна представляться как одно кольцо.

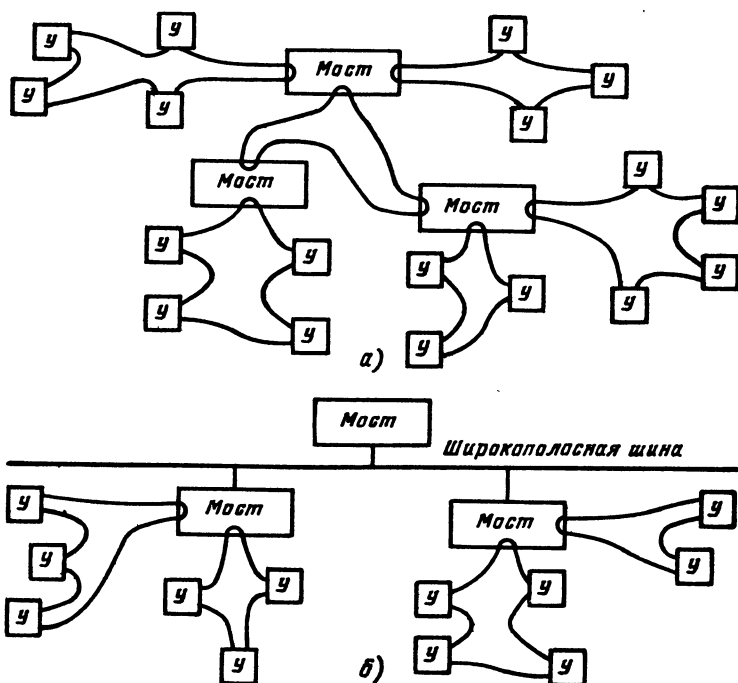


Рис. 16.26. Конфигурации звездно-кольцевой маркерной интерсети с центром в виде мостового кольца (а) и широкополосной шины (б)

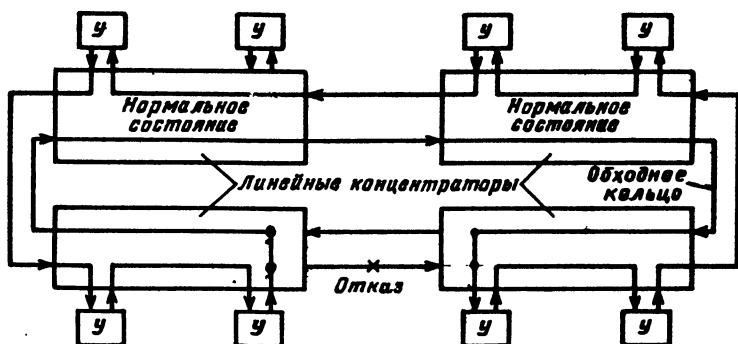


Рис. 16.27. Реконфигурация в кольцевой ЛВСт с маркерным доступом

В каждом кольце на сетевой адаптер одного из узлов возлагаются функции активного маркерного монитора (АММ), который должен реагировать на потерю маркера. Контроль потери маркера производится с помощью тайм-аута, устанавливаемого таймером, который АММ сбрасывает при каждом прохождении маркера.

Функции АММ может выполнять любой узел. При отказе АММ его функции берет на себя узел с наибольшим адресом.

Отказ в кольцевой сети вызывается разрывом сегмента кольца или неисправностью приемного или передающего элемента на концах сегмента. Узел, обнаруживающий потерю сигналов на своем приемнике, передает уникальную последовательность кадров («сигнальное состояние»), что приводит к выделению неисправного участка кольца.

Наличие линейных концентраторов облегчает удаление (обход) неисправных узлов из сети. Возможности реконфигурации значительно расширяются при наличии второго, альтернативного кольца с противоположным направлением передачи сигналов, связывающего все линейные концентраторы. В этом случае возможно отключение неисправного сегмента между двумя концентраторами без изменения логического порядка узлов в кольце, как показано на рис. 16.27.

Необходимые для реконфигурации переключения выполняются вручную или автоматически.

## **16.10. Принципы организации и средства телеобработки данных**

В автоматизированных системах планирования и управления рассредоточенными на значительной территории объектами, например промышленными предприятиями, транспортными процессами, при построении крупных информационно-вычислительных систем ЭВМ должна взаимодействовать с удаленными пользователями (абонентами). В таких случаях используются ВС с телеобработкой данных.

*Телеобработкой данных* называется режим работы вычислительной системы, при котором ЭВМ и удаленные абоненты [абонентские пункты (АП)] взаимодействуют через каналы передачи данных.

Вычислительные системы с телеобработкой данных можно рассматривать как простейший случай вычислительных сетей, в которых применяются лишь некоторые элементы сетевой технологии (например, коммутация каналов, но не коммутация пакетов).



Характерными режимами систем телеобработки данных являются:

а) дистанционные вычисления, при выполнении которых с АП через канал передачи данных вводятся в ЭВМ исходные данные и наименование запрашиваемой программы, а обратно на АП передаются результаты обработки программ;

б) дистанционный информационно-справочный режим, при реализации которого с АП через канал передачи данных посылается запрос на информацию, хранимую в базе данных ЭВМ, и запрашиваемая справка отсылается ЭВМ на пославший запрос АП;

в) дистанционный режим сбора данных, часто выполняемый путем предварительного нанесения подлежащих передаче сведений на перфоленту (перфокарты) с последующей передачей их из АП в ЭВМ через канал передачи данных при помощи перфоленточного (перфокарточного) устройства ввода;

г) системы коллективного пользования с дистанционным доступом (теледоступом) к ЭВМ абонентов с удаленных терминалов.

Структура системы телеобработки данных в ЕС ЭВМ представлена на рис. 16.28 [18].

Одним из абонентов системы телеобработки данных может быть другая ЭВМ. В таком случае получаем сеть телеобработки.

Реализация систем телеобработки данных требует специальных аппаратурных и программных средств.

Аппаратурные средства телеобработки включают в себя аппаратуру передачи данных (АПД), мультиплексоры передачи данных (МПД) и АП.

*Абонентский пункт* представляет собой объединяемый общим устройством управления и связи комплекс устройств ввода-вывода, с помощью которых абонент производит ввод данных в систему и получает сообщения от ЭВМ.

Функционирование комплекса оборудования АП организует-ся устройством управления и связи (УУС), которое устанавли-вает и прекращает связь АП с МПД, участвует в управлении передачей сообщений, воспринимая поступающие с линии и вы-давая в линию соответствующие управляющие сигналы, выпол-няет необходимые преобразования кодов, формирует контроль-ные суммы для передаваемых сообщений.

Оборудование большинства АП реализует как дистанцион-ную пакетную обработку, так и диалоговый режим работы. Некоторые АП позволяют организовать дистанционный удален-ный режим коллективного пользования ЭВМ для нескольких одновременно работающих пользователей.

*Аппаратура передачи данных.* В состав АПД входят модемы,

устройства защиты от ошибок, автоматические вызывные устройства (АВУ). В зависимости от скорости передачи информации и используемых каналов различают АПД низкоскоростную (до 200 бит/с по телеграфным каналам), среднескоростную (до 4800 бит/с по каналам тональной частоты) и высокоскоростную (более 4800 бит/с по широкополосным каналам).

Используются некоммутируемые (постоянно выделенные абоненту), а также коммутируемые каналы. В последнем случае связь между абонентскими пунктами и ЭВМ устанавливается автоматическими телефонными станциями (АТС) и станциями абонентского телеграфа (АТА). При этом в случае использования телефонных каналов со стороны ЭВМ вызов АП производится автоматически с помощью АВУ, а со стороны АП связь с ЭВМ устанавливается при помощи телефона. В случае использования телеграфных каналов для вызова со стороны ЭВМ и АП применяются вызывные приборы (ВП).

*Устройства защиты от ошибок (УЗО)* производят помехоустойчивое кодирование блоков данных при передаче и соответствующее декодирование при приеме, при этом используются циклические коды.

Если при приеме обнаруживается ошибка, то инициируется повторная передача ошибочного блока. При наличии УЗО достоверность передачи данных достигает  $10^{-7}$ .

*Мультиплексоры передачи данных* управляют передачей данных через сеть связи. В функции МПД входят реализация сопряжения между ЭВМ и аппаратурой передачи данных нескольких каналов, управление установлением и прекращением связи с несколькими АП, осуществление дисциплины обслуживания АП. Мультиплексоры подключаются к мультиплексному (байт-мультиплексному) каналу ЭВМ и управляются каналными программами.

Мультиплексоры передачи данных в процессе передачи данных для каждого канала связи осуществляют проверку правильности принятых блоков данных и их буферизацию, формируют контрольные символы при передаче блоков, производят преобразование кодов, формируют и распознают управляющие символы.

В ЕС ЭВМ при передаче данных по телеграфным каналам используется 5-разрядный код МТК-2, по телефонным и широкополосным каналам — 7-разрядный код КОИ-7 [22 а].

Мультиплексор передачи данных специального типа — «удаленный МПД» — выполняет помимо прочих также функции концентратора, связывая несколько низкоскоростных каналов связи с одним высокоскоростным.

Сложность выполняемых функций делает целесообразным использование в качестве МПД малых и микроЭВМ. В этом

случае имеется возможность в большей степени разгрузить основную ЭВМ от выполнения процедур, связанных с передачей данных по каналам связи, преобразованием кодов и др. Выполняющую такие функции ЭВМ называют связным процессором.

Программные средства телеобработки в ЕС ЭВМ представлены пакетами программ: а) *базисного телекоммутиационного метода доступа* (БТМД) и б) *общего телекоммутиационного метода доступа* (ОТМД), главным назначением которых является управление обменом информацией между ЭВМ и АП.

Пакет программ БТМД позволяет пользоваться специальными макрокомандами: *Открыть* и *Закрыть*, соответственно инициирующей и прекращающей сеанс связи через канал передачи данных; *Читать*, опрашивающей терминалы путем сканирования или по заданному адресу и в случае наличия подготовленного сообщения, принимающего его в ЭВМ; *Писать*, передающей сообщение адресуемому терминалу.

Программы этого пакета производят также необходимые преобразования кодов, контроль правильности передачи с помощью контролирующих кодов, диагностирование терминалов, сбор и обработку статистических данных об ошибках при передаче данных, динамическое распределение памяти при приеме сообщений. Метод теледоступа БТМД не обеспечивает телеобработку с очередями сообщений.

Более сложный пакет теледоступа ОТМД помимо функций, обеспечиваемых пакетом БТМД, позволяет также создавать очереди сообщений, обрабатывать сообщения с учетом их приоритетов и реализовывать некоторые другие возможности, например использовать язык пакета для упрощения составления программ специфических процедур телеобработки.

## Контрольные вопросы

1. Сравните по назначению, топологии, применяемым методам и средствам передачи данных глобальные (региональные) и локальные вычислительные сети.

2. Что означает утверждение: коммутация каналов обладает временной прозрачностью? Почему коммутация пакетов предпочтительна при реализации диалоговых режимов?

3. Поясните связь между развитием персональных компьютеров и развитием локальных вычислительных сетей.

4. В чем различие между способами передачи данных в интерфейсе ввода-вывода ЭВМ и в локальных вычислительных сетях?

5. Как обнаруживаются конфликты в шинных локальных сетях с множественным доступом с контролем несущей и обнаружением конфликтов (CSMA/CD)?

6. Как связаны назначение и особенности применения локальных сетей MAP и TOP с принятыми в них архитектурными решениями (топология, передающая среда, способ передачи информации, метод доступа и др.)?

7. Как и почему в локальных сетях (сетях с моноканалом) модифицируется многоуровневая эталонная модель вычислительных сетей?



## Список литературы

1. Антонюк Б. Л. МАП — Интегрированная система автоматизации производства // Микропроцессорные средства и системы. 1987. № 2. С. 37—40.
2. Авен О. И., Гурин Н. Н., Коган Я. А. Оценка качества и оптимизация вычислительных систем. М.: Наука, 1982.
3. Адасько В. И., Каган Б. М., Пац В. Б. Основы проектирования запоминающих устройств большей емкости. М.: Энергоатомиздат, 1984.
4. Алексеенко А. Г., Шагурин В. И. Микросхемотехника. М.: Радио и связь, 1982.
5. Альянах И. А. Внешние запоминающие устройства ЕС ЭВМ. М.: Советское радио, 1979.
6. Архитектура многопроцессорных вычислительных систем/О. С. Козлов, Е. А. Метлицкий, А. В. Экало и др./Под ред. В. И. Тимохина. Л.: Изд-во ЛГУ, 1981.
7. Баранов С. И. Синтез микропрограммных автоматов. Л.: Энергия, 1974.
8. Бендер Э., Гринберг К. Микропроцессор 80386 фирмы Intel // В мире персональных компьютеров. 1988. № 2. С. 18—26.
9. Богуславский Л. Б. Управление потоками данных в сетях ЭВМ. М.: Энергоатомиздат, 1984.
10. Богуславский Л. Б., Коган Я. А. Аналитическое исследование алгоритмов замещение страниц в двухуровневой памяти ЦВМ // Автоматика и телемеханика. 1974. № 11. С. 129—136.
11. Брябрин В. М. Программное обеспечение персональных ЭВМ. М.: Наука, 1988.
12. Высокоскоростные вычисления. Архитектура, производительность, прикладные алгоритмы и программы суперЭВМ: Пер. с англ./Под ред. Я. Ковалика. М.: Радио и связь, 1988.
13. Вычислительные сети и сетевые протоколы/Д. Девис, Д. Барбер, У. Прайс, С. Соломонидес. М.: Мир, 1982.
14. Ги К. Введение и локальные вычислительные сети. М.: Радио и связь, 1986.
15. Глушков В. М. Синтез цифровых автоматов. М.: Физматгиз, 1962.
16. Головкин Б. А. Параллельные вычислительные системы. М.: Наука, 1980.
17. Грицевский П. М., Мамченко А. Е., Степенский Б. М. Основы автоматики, импульсной и вычислительной техники. М.: Советское радио, 1979.
18. Дроздов Е. А., Комарницкий В. А., Пятибратов А. П. Электронные вычислительные машины единой системы. М.: Машиностроение, 1981.
19. Дружинин Г. В. Надежность автоматизированных производственных систем. М.: Энергоатомиздат, 1986.

20. Дудников Е. Е. Персональные компьютеры. М.: МЦНТИ, 1988.
21. Иванов Е. Л., Степанов И. М., Хомяков К. С. Периферийные устройства ЭВМ и систем: Учебник для вузов. М.: Высшая школа, 1987.
22. Каган Б. М. Электронные вычислительные машины и системы: Учеб. пособие для вузов.— 2-е изд., перераб. и доп. М.: Энергоатомиздат, 1985.
- 22 а. Каган Б. М. Электронные вычислительные машины и системы: Учеб. пособие для вузов. М.: Энергия, 1979.
23. Каган Б. М., Каневский М. М. Цифровые вычислительные машины и системы: Учеб. пособие для вузов/Под ред. Б. М. Кагана.— 2-е изд., перераб. М.: Энергия, 1973.
24. Каган Б. М., Воителев А. И., Лукьянов Л. М. Системы связи УВМ с объектом управления в АСУ ТП. М.: Советское радио, 1977.
25. Каган Б. М., Сташин В. В. Основы проектирования микропроцессорных устройств автоматики. М.: Энергоатомиздат, 1987.
26. Каган Б. М., Крейнин А. Я. Модели конфликтов в памяти мультипроцессорных систем // Автоматика и вычислительная техника. 1982. № 2. С. 59—65.
27. Каган Б. М., Мкртумян И. Б. Основы эксплуатации ЭВМ: Учеб. пособие для вузов. Под ред. Б. М. Кагана.— 2-е изд., перераб. и доп. М.: Энергоатомиздат, 1988.
28. Каган Б. М., Першеев В. Г. Систематизация структур адресных ЗУ // Микропроцессорные устройства и системы. М.: МИИТ. 1984. Вып. 710. С. 31—41.
29. Карцев М. А. Архитектура цифровых вычислительных систем. М.: Наука, 1978.
30. Клейнрок Л. Вычислительные системы с очередями. М.: Мир, 1979.
31. Коугли П. М. Архитектура конвейерных ЭВМ: Пер. с англ. М.: Радио и связь, 1985.
32. Ларионов А. М., Майоров С. А., Новиков Г. И. Вычислительные комплексы, системы, сети: Учебник для вузов. Л.: Энергоатомиздат. Ленингр. отд-ние, 1987.
33. Ломов Ю. С. ЭВМ высокой производительности ЕС 1066, ЕС 1065//ЭВТ. 1987. Вып. 1. С. 177—188.
34. Майерс Г. Архитектура современных ЭВМ: В 2-х кн. М.: Мир, 1985.
35. Лю Ю-Чжен, Чибсон Г. Микропроцессоры семейства 8086/8088. М.: Радио и связь, 1987.
36. Мельников В. А., Дадаев Ю. Г. СуперЭВМ: Проблемы создания, использования и развития // Вестник АН СССР. 1985. № 1. С. 56—69.
37. Мизин И. А., Богатырев В. А., Кулешов А. П. Сети коммутации пакетов/Под ред. В. С. Семенихина. М.: Радио и связь, 1986.
38. Микропроцессоры: В 3-х кн.: Учебник для вузов/Под ред. Л. Н. Преснухина. М.: Высшая школа, 1986.
39. Михновский С. Д., Шор Н. З. Оценка минимального числа пересылок при динамическом распределении страничной памяти // Кибернетика. 1965. № 5. С. 30—40.
40. Мячев А. А. Организация ввода-вывода. М.: Энергия, 1983.
41. Негеле Т., Янг Дж. Почему промышленность хорошо встречает новую серию персональных систем корпорации IBM // Электроника. 1987. № 8. С. 52—55.
42. Основы теории вычислительных систем: Учеб. пособие для вузов/Под ред. С. А. Майорова. М.: Высшая школа, 1978.
43. Пархоменко П. П., Согомонян Е. С. Основы технической диагностики. М.: Энергоиздат, 1981.

44. **Персональные компьютеры Единой системы ЭВМ/А. П. Запольский, В. Я. Пыхтин, А. Н. Чистяков, В. Б. Шкляр.** М.: Финансы и статистика, 1988.

45. **Перспективы развития вычислительной техники:** В 11 кн.: Справ. пособие/Под ред. Ю. М. Смирнова. Кн. 3: ЭВМ общего назначения/Ю. С. Ломов, К. С. Ораевский, А. П. Заморин, А. И. Слуцкий. М.: Высшая школа, 1989.

46. **Прангишвили И. В., Виленкин С. Я., Медведев И. Л.** Параллельные вычислительные системы с общим управлением. М.: Энергоатомиздат, 1983.

47. **Преснухин Л. Н., Нестеров П. В.** Цифровые вычислительные машины: Учеб. пособие для вузов. М.: Высшая школа, 1981.

48. **Пржиялковский В. В.** Состояние и проблемы развития ЭВМ общего назначения // ЭВТ. 1987. Вып. 1. С. 5—11.

49. **Пржиялковский В. В., Ломов Ю. С.** Технические и программные средства Единой системы ЭВМ (ЕС ЭВМ-2). М.: Статистика, 1980.

50. **Применение интегральных микросхем в электронной вычислительной технике:** Справочник/Под ред. Б. Н. Файзулаева, Б. В. Тарабрина. М.: Радио и связь, 1987.

51. **Проблемы создания управляющих комплексов повышенной живучести.**— Доклад на Всемирном электротехническом конгрессе/В. М. Долкорт, Б. М. Каган, М. М. Каневский и др. М.: ВЭЛК. 1977.

52. **Прохоров Н. Л.** Особенности архитектуры и программного обеспечения вычислительного комплекса СМ 1700 // Микропроцессорные средства и системы. 1988. № 2. С. 6—9.

53. **Савельев А. Я.** Прикладная теория цифровых автоматов: Учебник для вузов. М.: Высшая школа, 1987.

54. **Самофалов К. Г., Корнейчук В. И., Тарасенко В. П.** Цифровые электронные вычислительные машины: Учеб. пособие для вузов. Киев: Вища школа, 1983.

55. **Системы параллельной обработки:** Пер. с англ./Под ред. Д. Ивенса. М.: Мир, 1985.

56. **Смолов В. Б., Барашенко В. В., Байков В. Д.** Специализированные ЭВМ: Учебник для вузов/Под ред. В. Б. Смолова. М.: Высшая школа, 1981.

57. **Соловьев Г. Н., Никитин В. Д.** Операционные системы ЭВМ: Учеб. пособие для вузов. М.: Высшая школа, 1989.

58. **Сталлингс У.** Архитектура компьютера с сокращенным набором команд // ТИИЭР. 1988. № 1. С. 42—63.

59. **Сурков Л. В.** Проектирование архитектуры вычислительных сетей с использованием системы моделирования GPSS. М.: Изд-во МВТУ, 1987.

60. **Трейстер Р.** Персональный компьютер фирмы ИВМ. М.: Мир, 1986.

61. **Уэнсли Д. Х.** Высоконадежная система с тройным резервированием для управления технологическими процессами // Электроника. 1983. № 2. С. 32—39.

62. **Флинт Д.** Локальные сети ЭВМ: архитектура, принципиальное построение, реализация. М.: Финансы и статистика, 1986.

63. **Хендри Г.** Полностью аппаратное резервирование без простоя программ // Электроника. 1983. № 2. С. 39—43.

64. **Хуан К.** Перспективные методы параллельной обработки и архитектура суперЭВМ // ТИИЭР. 1987. № 10. С. 4—17.

65. **Четвериков В. Н.** Подготовка и телеобработка данных в АСУ: Учебник для вузов. М.: Высшая школа, 1981.

66. **Электронная вычислительная машина ЕС-1050** // Под общ. ред. А. М. Ларионова. М.: Статистика, 1976.

67. **Электронная** вычислительная машина ЕС-1045/А. Т. Кучукян, Т. Е. Саркисян, И. Б. Мкртумян и др. М.: Финансы и статистика, 1981.
68. **Электронная** вычислительная машина ЕС-1046/А. Т. Кучукян, Т. Е. Саркисян, Г. О. Патваканян и др. М.: Радио и связь, 1987.
69. **Якубайтис Э. А.** Архитектура вычислительных сетей. М.: Статистика, 1980.
70. **Cormier R. L., Dugan R. J., Guyette R. R.** System/370. Extended Architecture: The Channel Subsystem/IBM J. Res. Develop. 1983. № 3. P. 206—218.
71. **Data Processing-Open Systems Interconnection — Basis Reference Model** // ISO/DP 7498. August, 1981.
72. **Daves S.** Why the Q-bus lives on // Microprocessors and Microsystem. 1986. № 2. P. 109—114.
73. **Dixon R. C., Strole N. C., Markov J. D.** A token-ring network for local data communications // IBM System Journal. 1983. Vol. 22, № 1—2. P. 47—60.
74. **Doran K. W.** Computer Architective: A Structured Approach. London: Academic Press, 1979.
75. **Ercegovac M. D., Lang T.** Digital Systems and Hardware/Firmware Algorithms. N. Y.: Wiley, 1985.
76. **Falkoff A. D., Iverson K. E.** A formal description of System 360 // IBM System Journal. 1963. Vol. 3. № 3. P. 19—24.
77. **Flynn M. I.** Some Computer Organisations and their Effectiveness // IEEE Transactions on Computers. 1972. Vol. 21(9). P. 948—960.
78. **Reliability, Availability and Serviceability of IBM Computer Systems: A Quarter Century of Progress**/M. Y. Hsiao, W. C. Carter, I. W. Thomas, W. R. Stringfellow // IBM Journal Research and Development. 1981. № 3. P. 453—465.
79. **Leipold K., Spreen H.** Organisation des Nachrichtenverkehrs zwischen Zentraleinheiten und peripherien Einheiten in Datenverarbeitungssystem // Elektronische Rechenanlagen. 1969. № 3. S. 23—32.
80. **Muchmore S.** Multibus II message passing // Microprocessors and Microsystems. 1986. № 2. P. 91—93.
81. **Pittler M. S., Power D. M., Schnabel D. L.** System Development and Technology Aspects of the IBM 3081 Processor Complex // IBM Journal Research and Development. 1982. № 1. P. 2—11.
82. **Putman B. W.** Digital and Microprocessor Electronics: Theory, Applications, and Troubleshooting. Englewood Cliffs: Prentice Hall, 1986.
83. **Randell B.** The origins of digital computers — selected papers // Berlin: Springer-Verlag, 1973.
84. **Rauscher T. G., Adams P. M.** Microprogramming: A Tutorial and Survey of Recent Developments // IEEE Transactions on Computer. 1980. № 1. P. 2—19.
85. **Tabak D.** RISC systems // Microprocessors and Microsystems 1988. № 4. P. 179—184.
86. **Tucker S. G.** The IBM 3090 system: An overview // IBM System Journal. 1986. № 1. P. 4—19.
87. **Wilkes M. V.** The best way to design an automatic calculating machine // Report of Manchester University Computer. In Angular Conference. 1951. P. 16—18.
88. **Keating T. J., Prak J. W. L., Shoemaker K.** 32-bit microprocessor can run Unix and MS-BOS programs concurrently // Electron. Des. 1985. № 24. P. 115—122.

## АЛФАВИТНЫЙ УКАЗАТЕЛЬ

### А

- Автомат Мили 71, 219
  - Мура 71, 218
  - управляющий 163, 264
- Автоматический контроль 411
- Автоматическое восстановление вычислительного процесса 411, 422
  - диагностирование 410, 425
- Адаптер канал-канал 493
  - межшинной связи 502
- Адрес 9
- Адресации способы 237
- Адресная структура памяти 230
- Алгоритм деления чисел 190
  - замещения страниц 479
  - сложения двоичных чисел с фиксированной точкой 174
  - — вычитания чисел с плавающей точкой 196
  - умножения двоичных чисел 181
- АЛУ 9, 172, 174,
- Арбитр запросов прерывания 286
- Архитектура ЭВМ 15

### Б

- Байт 62
  - состояния 356, 378
- Байты уточненного состояния 356, 379
- Бит 52
- Блок операционный 163
  - управляющий 163
- Булева функция 72
- Буфер данных 369

### В

- Вектор прерывания 279
  - (слово) состояния программы 273
- Виртуальный канал 540
- Вычислительная сеть 41, 528
  - — локальная 532, 556
  - — — кольцевая 532, 558
  - — — с временными сегментами 564
  - — — — вставкой регистра 562
  - — — — — маркерным доступом 564
  - — — шинная 558
  - — — с доступом МДКН/ОК 565, 566
  - — — — — маркерным доступом 567
- Вычислительная система 38
  - — векторно-конвейерная 520
  - — коллективного пользования 40
  - — — — — многомашина 40, 489
  - — — — — многопроцессорная 40, 490
  - — — мультипрограммная 40
  - — — пакетной обработки 40
  - — — реального времени 41
  - — с коммутатором междоульных связей (Эльбрус) 516
  - — — общим потоком команд 514
  - — — — — управлением потоком данных 524
  - — — телеобработкой 41

## Д

Двоичная арифметика 46  
Дейтаграмма 540  
Дескриптор 255  
Дешифратор 88  
Дисплей 150  
— графический 154  
Дисциплина обслуживания 440

## Е

Единая система ЭВМ (ЕС ЭВМ)  
31, 33, 263, 289, 297, 377, 431,  
443

## З

Запоминающий массив 104  
— элемент 109  
ЗУ 99  
— биполярное 116  
— динамическое 118  
— на магнитных дисках гибких  
134, 145  
— — — жестких 133  
— — — — (типа Винчестер)  
134  
— — — лентах 131  
— оптические 145  
— постоянное (ПЗУ) 119

## И

Индексация 248  
Интерфейс ввода-вывода 346  
— — ЕС ЭВМ 377  
— «мультишина» 397  
— «мультишина» II (Multibus II)  
402  
— «общая шина» 389  
— «Q-шина» 286, 389

## К

Канал блокультиплексный 365  
— мультиплексный (блокультиплексный) 363, 366  
— селекторный 364, 370  
Канала адресное слово 357  
— управляющее слово 356, 358,  
360  
Канальная программа 360  
Квитирование 375  
Ключ защиты памяти 464

## Код ДКОИ 62

— дополнительный 49  
— обратный 48  
— прямой 48  
Команда передачи управления  
244  
— привилегированная 437  
Коммутация каналов 536  
— пакетов 538  
— сообщений 538  
Конвейер команд 302  
— операций 299  
Контроль арифметических операций 418  
— передач информации 412  
Корректирующий ход Хэмминга  
415  
Кэш-память 457

## М

Макропроцессор 28, 309, 320, 330,  
397, 402  
Мантисса числа 55  
Метод записи на магнитный носитель 135  
— ускоренная умножения 185  
Микрокоманда 164  
Микрооперация 164  
МикроЭВМ 28  
Мультиплексор 90  
«Мышь» 157

## О

Общее поле ОП 491  
Окно интерфейса 503  
Отказоустойчивый ВК 504

## П

Памяти динамическое распределение 469  
— емкость 100  
— цикл обращения 160, 101  
— ширина выборки 101  
Память 99  
— адресная 105  
— ассоциативная 106  
— виртуальная 469  
— ключей защиты 464  
— основная (оперативная) 103  
— сегментно-страничная 473

Память с последовательным доступом 102

— — прямым доступом 102

— стековая 108

— страничная 472

— управляющая 207, 212

Переключатель шины 502

Персональная система 343

Персональный компьютер (ПК)  
35, 143, 146, 156, 158, 162, 339

Печатающее устройство (принтер) 158

— — лазерное 161

— — матричное 159

— — «ромашка» 160

Поколение ЭВМ 15

Польская инверсная запись 242

Порт 29, 314

Порядок числа 55

— смещенный 57

Представление алфавитно-цифровой информации 62

— чисел десятичных 64

— — с плавающей точкой 56

— — фиксированной точкой  
54

Прерывание 274

— векторное 283

Префиксация 498

Принцип программного управления 11

хранимой в памяти программы 11

Программируемая логическая матрица 225

Программное (математическое) обеспечение 13

Протокол вычислительной сети  
543, 544

— канальный HDLC 549, 551

— X.25 548, 553

Процедура начальной выборки  
367, 381

— обслуживания ПУ 367, 383

Процессор 227

— матричный 446

## Р

Рабочий цикл процессора 246

Регистр 84

Рефреш 118

## С

Семафор 520

Система булевых функций 72

— — — функционально полная  
72

— виртуальных машин 450

— логических элементов 76

— малых ЭВМ (СМ ЭВМ) 31, 35,  
54, 231, 256, 258, 388, 500

Слово состояния канала 356

— — программы (процессора)  
243

СНК RISC-архитектура 269

Совместимость программная 32

Сумматор 93

СуперЭВМ 24

Схема комбинационная 71

Счетчик 86

## Т

Транспортная сеть 543

Триггер 77

ТЭГ 253

## У

Устройство ввода 11

— вывода 11

— периферийное (ПУ) 124

— управляющее 9, 163

## Ц

Цепочка данных 354

— операций 355

## Э

ЭВМ общего назначения 25, 482

— малая 26

— производительность 20

Элемент логический 73

Эталонная модель вычислительной сети 540

## Я

Язык микроопераций 167

## ОГЛАВЛЕНИЕ

Предисловие . . . . .	4
<b>Глава 1. Основные понятия . . . . .</b>	<b>7</b>
1.1. Два класса ЭВМ . . . . .	7
1.2. Принципы действия ЭВМ . . . . .	9
1.3. Понятие о системе программного (математического) обеспечения ЭВМ. Понятие об архитектуре ЭВМ . . . .	12
1.4. Поколения ЭВМ . . . . .	15
1.5. Основные характеристики ЭВМ . . . . .	20
1.6. Основные области применения вычислительной техники и основные типы ЭВМ . . . . .	23
1.6.1. СуперЭВМ . . . . .	24
1.6.2. ЭВМ общего назначения . . . . .	25
1.6.3. Малые ЭВМ . . . . .	26
1.6.4. Микропроцессоры и микроЭВМ . . . . .	28
1.6.5. МинисуперЭВМ и супермини-ЭВМ . . . . .	30
1.6.6. Специализированные ЭВМ . . . . .	31
1.7. Понятие о системах ЭВМ. Единая система ЭВМ общего назначения (ЕС ЭВМ) и система малых ЭВМ (СМ ЭВМ) . . . . .	31
1.8. Персональные компьютеры . . . . .	35
1.9. Классификация вычислительных систем . . . . .	37
<b>Глава 2. Представление информации в ЭВМ . . . . .</b>	<b>42</b>
2.1. Позиционные системы счисления . . . . .	42
2.2. Двоичная арифметика . . . . .	46
2.3. Прямой, обратный и дополнительный коды . . . . .	47
2.4. Формы представления чисел в ЭВМ . . . . .	52
2.5. Кодирование десятичных чисел и алфавитно-цифровой информации . . . . .	61
<b>Глава 3. Логические элементы и типовые узлы . . . . .</b>	<b>66</b>
3.1. Представление информации физическими сигналами . . . . .	66
3.2. Понятие о комбинационной схеме и цифровом автомате . . . . .	69
3.3. Системы логических элементов . . . . .	75
3.4. Триггеры . . . . .	77
3.5. Регистры . . . . .	84
3.6. Счетчики . . . . .	86
3.7. Дешифраторы . . . . .	88
3.8. Мультиплексоры . . . . .	90
3.9. Сумматоры . . . . .	93
3.10. Матричные БИС . . . . .	96



<b>Глава 4. Принцип построения устройств памяти . . . . .</b>	<b>99</b>
4.1. Общие сведения и классификация устройств памяти . . . . .	99
4.2. Адресная, ассоциативная и стековая организации памяти . . . . .	104
4.3. Структуры адресных ЗУ . . . . .	109
4.4. Запоминающие устройства с произвольным обращением . . . . .	116
4.5. Постоянные ЗУ . . . . .	119
<b>Глава 5. Периферийные устройства ЭВМ: внешние ЗУ и устройства ввода-вывода . . . . .</b>	<b>124</b>
5.1. Общие понятия о периферийных устройствах . . . . .	124
5.2. Принципы действия внешних ЗУ . . . . .	127
5.3. Методы записи информации на магнитный носитель . . . . .	135
5.4. Представление информации на магнитных лентах и дисках (ЕС ЭВМ) . . . . .	138
5.5. Периферийные устройства персональных компьютеров . . . . .	143
5.5.1. ЗУ на гибких и жестких магнитных дисках персональных компьютеров . . . . .	143
5.5.2. Клавиатура . . . . .	146
5.5.3. Дисплеи . . . . .	150
5.5.4. Средства управления изображением на экране дисплея персонального компьютера . . . . .	156
5.5.5. Печатающие устройства (принтеры) . . . . .	158
5.5.6. Воспроизведение звуков и музыки в ПК . . . . .	162
<b>Глава 6. Язык микроопераций . . . . .</b>	<b>163</b>
6.1. Декомпозиция вычислительного устройства на операционный и управляющий блоки. Принцип акад. В. М. Глушкова . . . . .	163
6.2. Иерархия языков описания вычислительных устройств . . . . .	164
6.3. Язык микроопераций . . . . .	167
<b>Глава 7. Принципы организации арифметическо-логических устройств . . . . .</b>	<b>172</b>
7.1. Общие сведения . . . . .	172
7.2. Структура и микропрограмма АЛУ для сложения и вычитания чисел с фиксированной точкой . . . . .	174
7.3. Структуры и микропрограмма АЛУ для умножения чисел с фиксированной точкой . . . . .	177
7.4. Методы ускорения умножения . . . . .	185
7.5. Структура АЛУ для деления чисел с фиксированной точкой . . . . .	187
7.6. Устройства для выполнения логических операций . . . . .	192
7.7. Особенности операций десятичной арифметики . . . . .	194
7.8. Операции над числами с плавающей точкой . . . . .	196
7.9. Многофункциональное АЛУ . . . . .	199
7.10. Особенности АЛУ микропроцессоров . . . . .	201
<b>Глава 8. Управляющие автоматы . . . . .</b>	<b>204</b>
8.1. Общие сведения . . . . .	204
8.2. Принцип действия управляющего автомата с хранимой в памяти логикой. Микропрограммное управление . . . . .	207
8.3. Управляющие автоматы с «жесткой» логикой . . . . .	215
8.4. Программируемые логические матрицы в управляющих автоматах . . . . .	225

<b>Глава 9. Процессоры и микропроцессоры: элементы архитектуры . . . . .</b>	<b>227</b>
9.1. Предварительные замечания . . . . .	227
9.2. Назначение и структура процессора . . . . .	227
9.3. Адресные структуры основных памятей . . . . .	230
9.4. Проблема выбора структуры и формата команд. Кодирование команд . . . . .	232
9.5. Способы адресации . . . . .	237
9.6. Стековая адресация . . . . .	242
9.7. Команды, процедуры и микропрограммы передачи управления в программах . . . . .	244
9.8. Индексация . . . . .	248
9.9. Теги и дескрипторы. Самоопределяемые данные . . . . .	253
9.10. Сопоставление программистских моделей (регистровых структур) машин общего назначения, малых и микроЭВМ и микропроцессоров . . . . .	256
9.11. Особенности адресации и системы команд 16-разрядных малых и микроЭВМ . . . . .	258
9.12. Структура команд ЕС ЭВМ . . . . .	263
9.13. Микропрограммная интерпретация языка команд ЭВМ . . . . .	265
9.14. Особенности RISC-архитектуры . . . . .	269
9.15. Понятие о состоянии процессора (программы). Вектор (слово) состояния . . . . .	272
9.16. Принципы организации системы прерывания программ . . . . .	274
9.17. Особенности систем прерывания малых ЭВМ . . . . .	285
9.18. Система прерывания и некоторые особенности организации режимов управления в ЕС ЭВМ . . . . .	289
9.19. Процедура выполнения команд. Рабочий цикл процессора . . . . .	296
9.20. Принцип совмещения операций академика С. А. Лебедева. Конвейер операций . . . . .	299
<b>Глава 10. Организация микропроцессоров и персональных компьютеров . . . . .</b>	<b>306</b>
10.1. Общие сведения о микропроцессорах . . . . .	306
10.2. Организация однокристальных 8-разрядных микропроцессоров . . . . .	309
10.3. Организация однокристальных 16-разрядных микропроцессоров . . . . .	320
10.4. Новые модели однокристальных 16- и 32-разрядных микропроцессоров (80286, 80386) . . . . .	330
10.5. Персональные компьютеры . . . . .	339
10.6. Новые персональные вычислительные средства — персональные системы . . . . .	343
<b>Глава 11. Принципы организации систем ввода-вывода. Интерфейсы ЭВМ и микропроцессоров . . . . .</b>	<b>345</b>
11.1. Проблемы организации систем ввода-вывода . . . . .	345
11.2. Прямой доступ к памяти . . . . .	347
11.3. Основные принципы построения и структуры системы ввода-вывода . . . . .	350
11.4. Основные функции каналов ввода-вывода. Управляющая информация для операций ввода-вывода . . . . .	354
11.5. Основные типы и структуры каналов ввода-вывода . . . . .	362

11.6.	Буферы данных в системах ввода-вывода . . . . .	369
11.7.	Элементы организации интерфейсов . . . . .	373
11.8.	Интерфейс ввода-вывода ЕС ЭВМ . . . . .	377
11.9.	Развитие системы ввода-вывода в ЕС ЭВМ . . . . .	384
11.10.	Структура и временные диаграммы интерфейса «Q-шина» малых ЭВМ . . . . .	388
11.11.	Интерфейс «мультишина» (И-41) микропроцессоров и микроЭВМ . . . . .	397
11.12.	Особенности интерфейса «мультишина-II» (Multi-bus II) . . . . .	402
 <b>Глава 12. Системы автоматического контроля и диагностирования ЭВМ . . . . .</b>		
12.1.	Основные характеристики надежности ЭВМ. Функции систем контроля и диагностирования . . . . .	407
12.2.	Контроль передачи информации . . . . .	412
12.3.	Контроль арифметических операций . . . . .	418
12.4.	Взаимодействие систем автоматического контроля, восстановления вычислительного процесса и диагностирования . . . . .	422
12.5.	Принципы построения систем автоматического диагностирования ЭВМ . . . . .	425
 <b>Глава 13. Принципы организации мультипрограммных ЭВМ общего назначения . . . . .</b>		
13.1.	Предварительные замечания . . . . .	431
13.2.	Организация мультипрограммного режима работы ЭВМ . . . . .	432
13.3.	Структура и принципы действия ВС коллективного пользования (разделения времени) . . . . .	437
13.4.	Системные программные средства мультипрограммных ЭВМ общего назначения . . . . .	443
13.5.	Особенности структуры ЭВМ общего назначения . . . . .	445
13.6.	Принцип виртуализации ресурса. Система виртуальных машин . . . . .	450
13.7.	Динамическое микропрограммирование — средство динамической модификации архитектуры ЭВМ . . . . .	453
 <b>Глава 14. Принципы организации многоуровневой системы памяти в мультипрограммных вычислительных системах . . . . .</b>		
14.1.	Проблемы организации памяти мультипрограммных систем . . . . .	456
14.2.	Согласование пропускных способностей процессора и памяти ЭВМ. КЭШ-память . . . . .	457
14.3.	Защита памяти . . . . .	462
14.4.	Организация работы памяти в режиме многоабонентного обслуживания . . . . .	466
14.5.	Динамическое распределение памяти. Организация виртуальной памяти . . . . .	469
14.6.	Алгоритмы управления многоуровневой памятью . . . . .	477
14.7.	Тенденции развития ЭВМ общего назначения . . . . .	482

<b>Глава 15. Принципы организации многопроцессорных и много- машинных вычислительных систем (комплексов) и суперЭВМ</b>	<b>489</b>
15.1. Понятие о многомашинных и многопроцессорных вычис- лительных системах и комплексах . . . . .	489
15.2. Методы и средства организации многомашинных и мно- гопроцессорных вычислительных комплексов на основе ЭВМ общего назначения (ЕС ЭВМ) . . . . .	493
15.3. Многопроцессорные и многомашинные комплексы с об- щей шиной (СМ ЭВМ) . . . . .	500
15.4. Особенности организации отказоустойчивых многопро- цессорных вычислительных комплексов . . . . .	504
15.5. Типы структур многопроцессорных ВС, ориентирован- ных на достижение сверхвысокой производительности	512
15.6. Многопроцессорная минисуперЭВМ с общим потоком команд (ПС-2000) . . . . .	514
15.7. Многопроцессорная суперЭВМ с коммутатором межмо- дульных связей («Эльбрус») . . . . .	516
15.8. Конвейерно-векторные суперЭВМ . . . . .	520
15.9. Концепция ВС с управлением потоком данных . . . . .	524
<b>Глава 16. Принципы организации вычислительных сетей</b>	<b>527</b>
16.1. Вычислительные сети. Общие сведения . . . . .	527
16.2. Классификация вычислительных сетей . . . . .	531
16.3. Методы передачи данных по каналам связи. Коммута- ция каналов, сообщений, пакетов . . . . .	533
16.4. Эталонная логическая модель вычислительной сети и иерархия протоколов . . . . .	540
16.5. Элементы протоколов . . . . .	544
16.6. Протоколы управления физическим и информационным каналами и сетью передачи данных. Протокол X.25	548
16.7. Локальные вычислительные сети. Основные понятия	556
16.8. Особенности организации передачи информации в ло- кальных сетях. Методы доступа к моноканалу . . . . .	560
16.9. Примеры локальных сетей: сеть TOP/MAP, сеть Ethernet, сеть IBM с маркерным доступом . . . . .	568
16.10. Принципы организации и средства телеобработки данных . . . . .	574
Список литературы . . . . .	580
Алфавитный указатель . . . . .	584

Учебное издание

Каган Борис Моисеевич

## **Электронные вычислительные машины и системы**

Редактор **В. Г. Першеев**

Редактор издательства **А. Н. Гусьяцкая**

Художественные редакторы **Т. А. Дворецкова,**

**Г. И. Панфилова**

Технические редакторы **О. Д. Кузнецова, В. В. Хапаева**

Корректор **Л. С. Тимохова**

ИБ № 2519

Сдано в набор 30.03.90. Подписано в печать 14.03.91. Формат 84 × 108<sup>1</sup>/<sub>32</sub>.  
Бумага газетная. Гарнитура литературная. Печать высокая. Усл. печ.  
л. 31,08. Усл. кр.-отт. 31,08. Уч.-изд. л. 38,59. Тираж 35 000 экз. Заказ 621.  
Цена 2 р.

Энергоатомиздат. 113114, Москва, М-114, Шлюзовая наб., 10

Ордена Октябрьской Революции, ордена Трудового Красного Знамени  
Ленинградское производственно-техническое объединение «Печатный  
Двор» имени А. М. Горького при Госкомпечати СССР. 197110, Ленин-  
град, П-110, Чкаловский пр., 15.

