ВЫСШЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

Н. Н. ГОРНЕЦ, А. Г. РОЩИН, В. В. СОЛОМЕНЦЕВ

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

Допущено
Учебно-методическим объединением вузов
по университетскому политехническому образованию
в качестве учебного пособия для студентов высших учебных заведений,
обучающихся по направлению подготовки
230100 «Информатика и вычислительная техника»

Москва
ACADEMIA
2006

УДК 681.3(075.8)
ББК 32.81я73
Г697

Рецензенты:
зав. кафедрой «Программное обеспечение ЭВМ и информационные технологии» Московского государственного технического университета им. Н. Э. Баумана, д-р техн. наук, проф. *Б. Г. Трусов*;
зав. кафедрой МГИЭиМ (Технический университет),
д-р техн. наук, проф. *В. С. Жданов*

Горнец Н.Н.

Г697 Организация ЭВМ и систем: учеб. пособие для студ. высш. учеб. заведений / Н. Н. Горнец, А. Г. Рощин, В. В. Соломенцев. — М. : Издательский центр «Академия», 2006. — 320 с.
ISBN 5-7695-2269-0

Рассмотрены основы теории построения вычислительных машин, принципы организации микропроцессоров, персональных компьютеров и многопроцессорных вычислительных систем. Приведены показатели их быстродействия и производительности. Изложены тенденции развития архитектур как персональных, так и многопроцессорных ЭВМ. Даны схемы наиболее распространенных периферийных устройств и методы сопряжения их с центральной частью машины.

Для студентов высших учебных заведений.

УДК 681.3(075.8)
ББК 32.81я73

ISBN 5-7695-2269-0

ПРЕДИСЛОВИЕ

В настоящем учебном пособии рассмотрены работа компьютера в целом, состав, назначение и принципы действия основных его компонентов, а также структура процессора, организация памяти и работа средств сопряжения основных устройств между собой, т.е. работа интерфейсов. Приведены некоторые способы повышения производительности и надежности путем объединения нескольких процессоров или машин в многопроцессорные и многомашинные системы, способы объединения машин в локальные сети и обеспечения надежности аппаратуры для работы компьютеров в такого рода сетях. Конечно, большие многопроцессорные и многомашинные компьютерные системы никак нельзя назвать «персональными». Они предназначены для обеспечения работы многих пользователей одновременно и требуют особых средств и специальных режимов. Нельзя не учитывать важность систем, решающих сложнейшие задачи, например прогнозирование погоды. Число таких систем от года к году не уменьшается.

Основное внимание в учебном пособии уделено построению аппаратуры, а не разработке программных средств. В то же время необходимо помнить, что без программ вычислительная машина представляет собой «кучу бесполезного железа»; ошибки в программном обеспечении столь же существенны, как и ошибки при разработке аппаратуры. Неудачно составленная программа может выполняться очень медленно, при этом затрачиваются огромные вычислительные ресурсы. Однако и ошибки при разработке аппаратуры недопустимы: примером могут служить первые модели процессора Pentium, в которых была обнаружена ошибка, приводившая к неправильным результатам при выполнении некоторых операций над числами с плавающей точкой.

В основу учебного пособия положен опыт преподавания дисциплин «Организация ЭВМ, комплексов и систем», «Вычислительные системы», «Периферийные устройства ЭВМ» в Московском государственном техническом университете гражданской авиации (МГТУГА).

Авторы выражают благодарность проф. МГТУГА В. Е. Смирнову за предоставленные материалы для гл. 7 «Интерфейсы».

ВВЕДЕНИЕ

Без вычислительных машин, или компьютеров, в настоящее время невозможна ни одна сфера человеческой деятельности. Компьютеры стали частью не только сферы производства, но и домашнего быта. Множество людей проводят часы за экраном своего компьютера, получая последние новости, биржевые сводки, цены, технические сведения, прогнозы погоды и многое другое из сети Интернет, а также используют компьютер для игр и развлечений. При написании этого учебного пособия авторы также широко пользовались компьютером и материалами из Интернета. Это позволило не только подготовить рукопись значительно быстрее, но и сделать ее полнее и современнее.

Термин «электронная вычислительная машина», или ЭВМ, совершенно не означает, что она предназначена только для каких-либо вычислений. Это уже не просто вычислительные машины, а системы обработки данных, способные хранить информацию, редактировать, обновлять, выполнять сортировку и поиск нужных данных, формировать таблицы, диаграммы и отчеты, осуществлять логические преобразования, выдачу результатов и т. п. По этой причине в настоящее время вычислительную машину (особенно персональную) принято называть английским термином «компьютер».

Повсеместное внедрение компьютеров в современную жизнь, регулярное обновление их аппаратных и программных средств, постоянная модернизация и появление все новых компонентов требуют глубокого знания принципов их работы. Аппаратные средства компьютеров, а именно современные процессоры, память, периферийные устройства и устройства подключения вычислительных машин к сетям, описаны в учебной литературе явно недостаточно. Возможно, это вызвано тем, что, покупая «готовый» компьютер, потребитель не пытается узнать, как он устроен. Больше всего покупателя интересуют стандартные программные средства и их возможности для удовлетворения своих потребностей. Но невозможно понять работу программных средств, не обладая хотя бы минимальными знаниями об аппаратуре.

Небольшие по габаритным размерам, но обладающие достаточной вычислительной мощностью компьютеры стали широко доступными. Объединение нескольких таких машин между собой, т. е. создание компьютерных сетей, обеспечило получение самой

разнообразной информации. В настоящее время компьютерами пользуется все цивилизованное человечество, это стало возможным благодаря дружественым программам и интерфейсам, ставшим непременной частью компьютера. Получила развитие всемирная сеть Интернет, объединившая средства вычислительной техники и техники связи и дающая возможность узнавать и сообщать последние новости. Теперь практически все машины тем или иным способом взаимодействуют с глобальными и локальными сетями, т. е. компьютеры становятся компонентами все более крупных систем.

Для подключения к такой сети не требуется сложных и дорогостоящих аппаратных средств — достаточно простых и недорогих адаптеров, модемов и каналов передачи данных. При этом скорость получения информации будет определяться быстродействием самой машины и пропускной способностью каналов связи, работой программ, размещением и надежностью всех компонентов, участвующих в доставке информации пользователю.

ГЛАВА 1

СТРУКТУРА СОВРЕМЕННОГО КОМПЬЮТЕРА

1.1. Основные понятия

Компьютер — это устройство, предназначенное для обработки и преобразования информации. Долгое время его называли электронной вычислительной машиной (ЭВМ), цифровой вычислительной машиной (ЦВМ) или электронной цифровой вычислительной машиной (ЭЦВМ).

Первая электронная цифровая вычислительная машина, или программируемый калькулятор ENIAC — Electronic Numerical Integrator and Computer, — была создана в Пенсильванском университете под руководством Д. Мочли и П. Эккерта в 1945 г. На роль первой машины претендуют также разработанный в начале 1940-х гг. Дж. Атанасовым и К. Берри специализированный калькулятор ABC и предназначенный для расшифровки кодов немецкой шифровальной машины вычислитель Colossus, созданный под руководством М. Ньюмена. В 1951 г. под руководством С. А. Лебедева была создана первая советская машина — малая электронная счетная машина (МЭСМ), а в начале 1953 г. — большая (БЭСМ), быстродействие которой оценивалось в 8000 операций/с. Все эти машины были ламповыми и впоследствии были отнесены к ЭВМ первого поколения. Этот период ознаменовался поиском инженерных решений для построения различных устройств.

Появление транзисторов позволило значительно усложнить структуру машин, относимых ко второму поколению. Появились так называемые индексные регистры, упрощающие доступ к массивам данных. В качестве оперативной начали использовать память на ферритовых сердечниках. Управление вводом-выводом возлагалось на отдельные блоки, что позволило выполнять загрузку данных одновременно с арифметическими операциями, но потребовало специальных средств для синхронизации процессов. В этот же период были созданы первые языки программирования высокого уровня: Фортран, Алгол, Кобол. К машинам второго поколения относятся советские машины «Урал», «Минск-22», «Минск-32», БЭСМ-2 и др.

Интегральные схемы позволили еще больше усложнить машину и увеличить ее мощность. Получило распространение конвей-

ерное выполнение команд, начала применяться параллельная обработка, в устройствах управления стали использовать принцип микропрограммирования. Появился стандартный интерфейс для подключения периферийных устройств. Именно в это время начали выпускать машины серии IBM/360 (370) и ЕС ЭВМ, с появлением которых и связан термин «поколение ЭВМ». В дальнейшем к ЭВМ третьего поколения стали относить любые машины, построенные на интегральных схемах малой и средней интеграции.

Машины на базе больших интегральных схем (БИС) и сверхбольших интегральных схем (СБИС) стали называть машинами четвертого поколения. Память машин начали строить на полупроводниковых элементах. В этот же период была разработана концепция машины с сокращенным набором команд (RISC). В 1980-х гг. появилась японская программа по созданию ЭВМ пятого поколения. Однако широкое распространение персональных компьютеров привело к падению интереса к «поколениям ЭВМ» и теперь этот термин выходит из употребления.

Сегодня компьютер стал устройством, способным хранить и обрабатывать огромное количество информации. В течение многих лет передача различных сведений производилась посредством устной речи, графики, рукописных или печатных символов, а обработка информации осуществлялась исключительно мозгом человека. С появлением компьютера эта монополия нарушилась, что потребовало создания средств для загрузки и выгрузки информации в его обрабатывающую часть. Были созданы периферийные устройства, предназначенные для преобразования информации (например, графического изображения в текст), кодирования (т.е. замены отдельных символов их кодовыми эквивалентами) и изменения формы представления кодированной информации (например, штрихи в комбинацию высоких и низких уровней потенциала). Это вызвано тем, что способы представления и обработки информации в компьютере отличаются от тех, что используются мозгом человека и другими объектами внешнего мира.

Представленные в формализованном виде сведения часто называют *данными*, представление данных осуществляется квантами информации. Под *квантом информации* следует понимать некоторый ее объем, принятый при описании объекта, а также при передаче, хранении и обработке. Кванты информации различны для компьютера и внешнего мира.

В вычислительной технике часто используют и более узкое толкование термина «данные», служащее для различения обрабатываемой (исходные и конечные данные) и управляющей (команды, адреса) информации. Все обмены данными между различными устройствами компьютера, а также между компьютером и объектами внешнего мира осуществляются посредством сообщений. *Сообщение* — это произвольное количество информации, пред-

назначенное для передачи между различными устройствами, с явно или неявно указанными началом и концом.

Мы будем иметь дело в основном с дискретными сообщениями; в них данные представлены конечным числом квантов информации в виде последовательности символов из некоторого набора, называемого *алфавитом*. Алфавит может включать в себя цифры, обычные буквы, специальные символы, символы псевдографики и т. п. Все дискретные сообщения кодируются, но независимо от способа кодирования важнейшими элементами их представления являются биты, байты, машинные слова и файлы.

Бит (от *англ.* binary digit, или bit) — это наименьшая «порция» информации в двоичной системе. Он служит для внутреннего представления чисел и команд в машине и может принимать значения «0» и «1».

В конце 1960-х гг. компьютеры стали применять не только для вычислений, но и для обработки буквенно-цифровой информации. Тогда потребовалось кодировать не только десятичные цифры, но и буквы, знаки, различные специальные символы. Общее число используемых символов, подлежащих кодированию, в то время равнялось 256. Для их кодирования потребовался слог (часть машинного слова) из восьми бит; его и назвали *байтом*. Предназначенные исключительно для США машины IBM/360 использовали всего 128 разнообразных символов, поэтому в них слог (байт) состоял из семи бит. Однако расширение алфавита компьютера потребовало включить в него буквы национальных алфавитов, например русского. Так появился восьмибитовый байт. В памяти компьютера теперь хранилась буквенно-цифровая информация, а каждый символ кодировался в виде байта. Это привело к тому, что длину *машинного слова*, т.е. множество битов, рассматриваемых аппаратурой компьютера как единое целое, стали делать кратной байту. В одном машинном слове может размещаться команда или целое число. Длина машинного слова в настоящее время составляет 16, 32 или 64 бита (соответственно, 2, 4 или 8 байт). Однако переменный формат команд в большинстве персональных компьютеров привел к тому, что термин «машинное слово» стал использоваться все реже.

Байты и машинные слова объединяются в *файлы*, имеющие определенную структуру. Обычно файл содержит название, некоторые характеристики, символы контроля и собственно команды (командный файл) или данные. Файл может иметь произвольный размер и быть представлен в нескольких форматах. Файл является не «машинной» единицей информации, а скорее логической. В зависимости от вида хранящейся в файлах информации различают командные файлы (программы), текстовые, графические и т. п.

1.2. Принцип действия компьютера

Чтобы понять принцип действия компьютера, остановимся подробнее на наиболее распространенной и простой структуре персонального компьютера, или ПЭВМ. Основное отличие персонального компьютера от больших машин, или так называемых мейнфреймов, состоит в том, что он позволяет одновременно использовать его ресурсы только одному пользователю. Казалось бы, что такой компьютер должен работать исключительно в однопрограммном режиме, т.е. выполнять одну текущую программу, но это не так. Он может выполнять одновременно несколько программ: обработки, вывода результатов, загрузки, поиска информации в сети и т.д. Кроме того, многие персональные машины используются в качестве серверов в сети, и их ресурсами (т.е. аппаратными и программными средствами) могут пользоваться несколько пользователей одновременно.

Структура самого компьютера за все время существования машин изменилась незначительно. Она по-прежнему строится на основе модели фон Неймана, во всяком случае, ее основная память состоит из отдельных ячеек с последовательными номерами (или «адресами»), в которых могут храниться как коды отдельных команд (программа), так и данных. Однако технологический прогресс привел к объединению нескольких узлов и устройств в одной микросхеме.

Упрощенная структура компьютера (рис. 1.1) состоит из следующих основных узлов: арифметико-логическое устройство (АЛУ), оперативное запоминающее устройство (ОЗУ), управляющее устройство (УУ), устройство ввода данных в машину (УВв) и устройство вывода результатов проведенных расчетов (УВыв). Именно такую «пятиблочную» структуру имели вычислительные машины первого поколения. Помимо перечисленных узлов любой компьютер имеет пульт ручного управления, предназначенный для включения машины и слежения за правильностью ее работы.

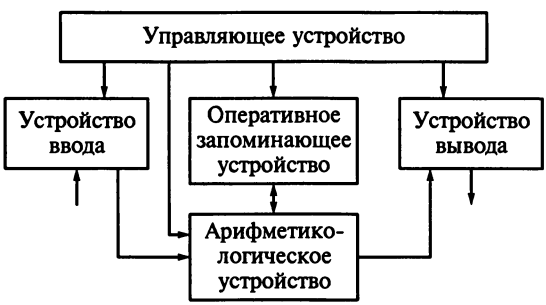


Рис. 1.1. Упрощенная структура компьютера

Теперь принято называть АЛУ с соответствующими схемами управления *процессором*, схемы для управления и подключения периферийных устройств — *контроллерами* и *адаптерами*, а передача информации между блоками компьютера осуществляется по *шинам интерфейса*. Арифметико-логическое устройство предназначено для выполнения арифметических и логических операций над машинными словами, т.е. кодами, находящимися в памяти и поступающими в АЛУ для обработки. Кроме того, оно выполняет различные операции по управлению вычислениями.

Оперативное запоминающее устройство, или оперативная память, хранит коды машинных слов (команд и данных) в своих ячейках. Эти ячейки нумеруются, а номер ячейки называется *адресом*. В памяти компьютера, как правило, находятся только команды и данные. Машина использует хранимую в ОЗУ информацию для организации вычислительного процесса. Информация попадает в ОЗУ из устройства ввода или из внешнего запоминающего устройства (ВЗУ). Внешняя память позволяет хранить большие объемы информации, но обладает меньшим быстродействием по сравнению с ОЗУ. В течение всего процесса обработки информация поступает в АЛУ только из ОЗУ, а результаты выполнения программы выдаются на устройство вывода после окончания обработки. Точно так же информация из ВЗУ, прежде чем принять участие в обработке, должна быть предварительно переписана в ОЗУ.

Устройство управления служит для автоматического управления вычислительным процессом; оно формирует сигналы управления на все устройства компьютера, преобразуя команды программы в управляющие сигналы. Если узел управления совмещен с АЛУ, то такое объединенное устройство называют центральным процессором (ЦП). Он связан с основной памятью (ОП), состоящей из ОЗУ и постоянного запоминающего устройства (ПЗУ), или постоянной памяти, предназначенной для хранения программ ввода-вывода, и различными устройствами ввода и вывода (или периферийными устройствами) посредством шины (рис. 1.2), называемой часто общей шиной (ОШ). Такая общая шина состоит из нескольких «подшин»: адреса, данных и управления. Мы будем их называть просто шинами. В персональных машинах для экономии места на системной плате (т.е. плате, на которой расположены процессор, память и разъемы для подключения периферийных устройств) шины адреса и данных иногда выполняют в виде одной разделяемой во времени шины; тогда адрес и данные по ней передаются только поочередно.

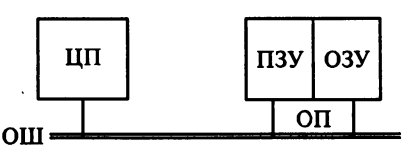


Рис. 1.2. Центральная часть машины

Помимо ЦП и ОП компьютер содержит множество периферийных (или внешних) устройств, предназначенных для связи с внешним миром (человеком, объектами управления и т.п.). Эти устройства подключаются к ОШ с помощью контроллеров, адаптеров, шинных мостов и т.п.

В персональном компьютере (а в последнее время и в компьютерах других типов) основная память состоит из двух частей — постоянной и оперативной. В очень небольшой по современным понятиям (она достигает нескольких мегабайт) постоянной памяти хранится программа начальной загрузки, называемая BIOS (Basic Input-Output System). Эта информация «зашита» в памяти, т.е. хранится постоянно. Оперативная память в момент включения компьютера не содержит никакой информации. При его включении на все блоки подается сигнал установки в исходное «нулевое» состояние, начинают формироваться тактовые импульсы и компьютер начинает работать.

Чтобы понять, как работает компьютер, нужно знать, из каких элементов он состоит, т.е. что такое триггер, счетчик, регистр, логическая схема и т.п. Подробнее о работе всех этих компонентов можно узнать из последующих разделов книги. Здесь же рассмотрим только основные понятия. *Триггер* представляет собой электронную схему, которая может находиться в одном из двух устойчивых состояний «0» и «1». Внешними сигналами можно переводить триггер из одного состояния в другое. *Регистр* — это несколько определенным образом соединенных триггеров, т.е. можно записать двоичное слово в регистр, прочитать его, сдвинуть, инвертировать. *Счетчик* позволяет определить число поступивших на него сигналов. Он также строится на основе триггеров. *Логическая схема* реализует определенную логическую функцию, т.е. формирует выходной сигнал при определенных комбинациях сигналов на ее входах.

Продолжим рассмотрение работы простейшего компьютера. Содержимое счетчика команд (СчК; его называют также instruction pointer — IP) процессора передается по адресной шине на регистр адреса (РгА) основной памяти (рис. 1.3). В момент включения компьютера в счетчике команд всегда находится один и тот же начальный адрес. Таким образом, запрашивается содержимое ячейки памяти с этим начальным адресом, принадлежащим BIOS. Как правило, эта ячейка содержит код команды безусловного перехода, служащей для изменения содержимого счетчика команд. Этот код передается на регистр команд (РгК) процессора по шине данных. Содержимое ячейки памяти поступает на РгК, поскольку запрос к памяти произведен из счетчика команд; это *обязательное требование* для любого компьютера традиционной архитектуры.

Регистр команд процессора, в свою очередь, состоит из нескольких регистров — регистра кода операции (РгКОП) и регис-

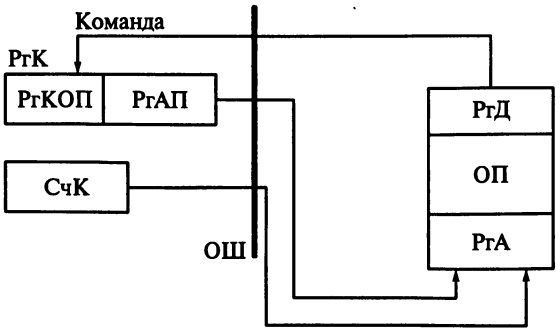


Рис. 1.3. Передача команд из ОП в ЦП

тров адресов процессора (РгАП). Часть слова (содержимого ячейки ОП, к которой произведено обращение), попавшая в регистр кода операции, передается в блок управления (БУ), вырабатывающий последовательность управляющих сигналов.

Когда выполняется команда безусловного перехода, вторая адресная часть слова, попавшая в один из регистров адреса процессора, под управлением сигналов с БУ передается вновь на счетчик команд. Эта команда одноадресная, т. е. ее адресная часть содержит только один адрес. На этом и завершается ее выполнение. Блок управления формирует сигнал об окончании выполнения команды, а содержимое СчК вновь передается на РгА памяти, т. е. происходит запрос следующей команды.

Таким образом, процедура обращения к памяти повторяется. Содержимое ячейки памяти, к которой произведено повторное обращение, рассматривается в качестве новой команды, т. е. вновь загружается на РгК процессора. Обычно вторая команда служит для начала загрузки ОЗУ с магнитного диска; она уже не является командой безусловного перехода. При ее выполнении под управлением кода операции (часть команды, попавшая на РгКОП) вырабатываются иные управляющие сигналы, а содержимое первого регистра РгАП, представляющего собой часть РгК, передается на адресный регистр памяти и рассматривается в качестве адреса первого операнда.

Для ОЗУ безразлично, откуда пришел запрос — из счетчика команд или адресного регистра, поэтому в регистре данных (РгД) памяти слово формируется так же, как и раньше. Однако в процессоре оно помещается на первый регистр данных АЛУ, поскольку запрос этого слова поступил из адресного регистра РгАП. Затем блок управления формирует аналогичные сигналы для передачи на РгА памяти содержимого второго РгАП; в результате содержимое ячейки памяти с адресом, находящимся в РгАП, поступает на второй регистр данных арифметического устройства.

Затем блок управления вырабатывает сигналы в зависимости от кода операции в РгКОП, подает их в АЛУ, которое выполняет соответствующую операцию, а ее результат помещает в выходной регистр-аккумулятор. После этого содержимое регистра-аккумулятора передается в ячейку памяти, адрес которой обычно находится в первом РгАП, т.е. выполняется еще одно обращение к ОП. Информация из регистра-аккумулятора передается на шину данных, а адрес ячейки из РгАП — на адресную шину. В зависимости от конструкции машины, числа адресов в выполняемой команде (адресности) и других особенностей, содержимое регистра-аккумулятора может сохраняться в нем, передаваться в ячейку ОП по адресу, находящемуся в первом или втором РгАП.

После сохранения содержимого регистра-аккумулятора к счетчику команд (СчК) добавляется длина текущей команды в байтах (часто говорят «единица»), чтобы обратиться к следующей ячейке памяти, и начинается новый цикл выполнения очередной команды.

Таким образом, выполнение программы происходит последовательно: каждый раз в машине реализуется лишь одна команда, попадающая в регистр команд из ОП. Чтобы увеличить производительность компьютера, нужно либо повысить скорость выполнения команды, либо выполнять несколько последовательных команд одновременно. Повышение скорости выполнения команды связано с улучшением технических характеристик и увеличением быстродействия всех компонентов, входящих в компьютер — ЦП, ОП, шин интерфейсов, устройств ввода-вывода. Но увеличение скорости выполнения команды принципиально ограничено — скорость распространения сигналов в машине не может превышать скорость света, а длина пути определяется числом вентилей и применяемой технологией. Второй путь, заключающийся в параллельном выполнении нескольких команд, наиболее перспективен. Однако и он обладает рядом ограничений, которые мы рассмотрим далее.

1.3. Программное обеспечение компьютера

Компьютер — это сложное сочетание аппаратных и программных средств, но пользователи обычно даже не заинтересованы в их понимании. Для них компьютер является инструментом, с помощью которого они намерены решить конкретную прикладную задачу, т.е. разработать и отладить программу, а затем ввести конкретные данные и выполнить ее. Программное обеспечение (ПО) призвано ускорить и упростить этот процесс, а также организовать более полное использование аппаратных средств компьютера. В настоящее время уже никто не пользуется «голой» ап-

паратурой, программируя задачу в кодах машины. Программу составляют на языке высокого уровня, например C++ или Паскале, а затем эта программа транслируется на внутренний язык компьютера.

Обычно программные средства условно делят на три группы: операционные системы, программы технического обслуживания и пакеты прикладных программ.

Вся работа компьютера осуществляется под управлением *операционной системы* (ОС) — комплекса программ, предназначенного для распределения ресурсов компьютера, управления режимами его работы, облегчения подготовки программ, организации их выполнения и общения пользователя с компьютером. В этот комплекс входят трансляторы с определенных языков программирования.

Все персональные компьютеры имеют трансляторы с Паскаля, C++ и некоторых других языков, а конвейерно-векторные — трансляторы с Фортрана.

При работе компьютера возникает множество задач по планированию работы процессора, распределению и защите памяти, управлению периферийными устройствами и т. п. Но пользователи не имеют прямого доступа к отдельным устройствам компьютера. Связь их с этими аппаратными средствами осуществляется также при помощи программ ОС.

Обычно ОС имеет определенный графический интерфейс, с помощью которого пользователь выбирает необходимые для него действия. В персональных компьютерах IBM PC таким графическим интерфейсом де-факто служит заставка одной из ОС Microsoft Windows. Даже если используются ОС Linux, Solaris или какая-либо иная, при включении компьютера в начале работы все равно появляется такая заставка.

Программы технического обслуживания предназначены для упрощения трудоемкости при эксплуатации компьютера. В состав таких средств входят программы проверки работоспособности компьютера (такие программы обязательно выполняются при включении компьютера в работу), диагностирования неисправностей, выявления имеющихся периферийных устройств и их состояния. В современных персональных компьютерах эти программы входят в состав базовой системы ввода-вывода BIOS и выполняются при включении компьютера в работу. (Программы технического обслуживания в мейнфреймах обычно представляют собой часть ОС.)

Пакеты прикладных программ — это комплекты программ, предназначенные для решения часто встречающихся классов задач и расширения функций ОС, например для управления базами данных. Наличие пакетов прикладных программ значительно упрощает и ускоряет процесс программирования.

1.4. Надежность, производительность, быстродействие и его показатели

Надежность. Отказы компьютера могут оказать критическое влияние на работу информационно-справочной системы или системы управления. Отказ компьютера в информационно-справочной системе приводит к задержке получения ответа, иногда весьма длительной, что может вызвать большие экономические потери. Задержка в выдаче управляющего воздействия в результате отказа компьютера приводит к потере управления, разрушению технологических установок, срыву выполнения задания, т.е. к значительным потерям, а в некоторых случаях и к экологическим катастрофам.

Современные технологии не позволяют полностью избавиться от отказов в аппаратуре и ошибок в программах даже при значительных затратах, но ряд мер технического и организационного характера способны снизить их интенсивность.

Надежность — это свойство компьютера сохранять свою работоспособность, т.е. выполнять возложенные на него функции. Однако в работе компьютера возможны ошибки, которые подразделяют на систематические, возникающие в результате отказов, и случайные, возникающие в результате сбоев. *Отказ* — это утрата возможности выполнения требуемой функции, а *сбой* — кратковременное нарушение правильной работы аппаратуры (может быть вызвано искрением, вибрацией, внешними помехами и т.п.).

Как правило, надежность принято характеризовать вероятностью безотказной работы.

Производительность. Способность машины выполнять некоторый объем работы по обработке данных за единицу времени называют *производительностью.* На нее оказывает влияние множество факторов: характер выполняемых задач, архитектура и параметры процессора, характеристики основной и внешней памяти, наличие дополнительных устройств обработки, быстродействие соединительных шин, способ подключения различных устройств и т.п. Оценка производительности проводится по стандартным методикам, служащим только для непосредственного сравнения различных компьютеров между собой.

Для оценки производительности часто используют понятие *времени прохождения задачи* (*времени ответа*, *времени выполнения*) — интервал от момента поступления задачи на выполнение до момента представления результатов ее решения пользователю. Это время включает в себя работу ЦП, обращения к оперативной памяти и дискам, операции ввода-вывода, накладные расходы, связанные с работой ОП, и т.п. В мультипрограммном режиме, когда процессор компьютера выполняет несколько программ, во время ожидания завершения процедуры ввода-вывода для какой-

либо программы ЦП может выполнять другую программу, поэтому время прохождения задачи не будет постоянным. Обычно производительность компьютера в целом характеризуется средним временем прохождения задачи.

Быстродействие и его показатели. Способность компьютера выполнять определенное число операций за единицу времени называют *быстродействием*. В большинстве случаев его можно определить по тактовой частоте генератора, которая стала важнейшей характеристикой быстродействия. При этом оценивается быстродействие только процессора. Время выполнения программы в ЦП зависит от трех параметров: длительности такта синхронизации (или тактовой частоты), числа тактов синхронизации, необходимого для выполнения каждой команды, и общего числа команд в программе. Для оценки быстродействия персональных компьютеров особенно часто используют частоту. Сегодня она достигает 3,5 ГГц и выше, а фирмы-изготовители, например Intel или AMD, продолжают ее наращивать.

Кроме того, быстродействие можно оценить по следующим показателям:

длительность выполнения операций определенного типа (обычно в качестве параметра быстродействия фирмы-производители приводят число наиболее коротких арифметических операций, выполняемых за секунду; это так называемое пиковое быстродействие процессора);

средняя длительность выполнения операции из некоторого стандартного набора операций, называемого смесью; это номинальное быстродействие;

средняя длительность выполнения представительной задачи; при этом если в затратах времени учитывается только время обработки, то такую задачу принято называть ядром, а если учитывается и время на ввод-вывод, то эталонной задачей, или бенчмарком. Время на организацию вычислительного процесса не учитывается. Это системная производительность.

Быстродействие компьютера, имеющего традиционную архитектуру и призванного решать задачи с большим числом логических операций, принято оценивать числом MIPS (миллион инструкций в секунду). Этот показатель прост для понимания — для более «быстрого» компьютера характерно более высокое значение этого показателя. Тем не менее применение MIPS в качестве универсального показателя наталкивается на ряд трудностей.

Во-первых, каждый компьютер обладает структурой, ориентированной на обработку слов определенного формата и разрядности, т.е. инструкции в разных компьютерах определяют различный объем работы.

Во-вторых, этот показатель не учитывает сложность выполняемой команды. Поэтому при наличии в компьютере дополнитель-

ного сопроцессора, призванного выполнять операции с плавающей точкой (ПТ), этот показатель снижается. При отсутствии сопроцессора операции над числами с ПТ реализуются посредством подпрограмм, состоящих из нескольких достаточно простых команд целочисленной арифметики, и показатель MIPS имеет высокое значение. Сложная команда сопроцессора выполняется достаточно долго (хотя и значительно быстрее, чем соответствующая подпрограмма), поэтому показатель MIPS снижается. Этот показатель имеет и ряд других недостатков.

Особый интерес вызывает оценка быстродействия компьютеров, призванных решать научно-технические задачи, в которых широко используется арифметика с ПТ. Обычно их быстродействие оценивают в миллионах операций с ПТ в секунду — MFLOPS. Многие программисты считают, что одна и та же программа, написанная для различных компьютеров, выполняет разное число команд, однако число операций над числами с ПТ в этих программах одинаково. Поэтому MFLOPS может служить для сравнения разных компьютеров между собой при выполнении одной и той же программы. Однако и этот показатель не универсален, поскольку в компьютерах используются разные системы (или наборы) команд. Таким образом, характеризовать машину единственным показателем невозможно без указания программы, выполняемой в компьютере.

В настоящее время разработано два набора тестов, предназначенных для измерения производительности процессора, системы памяти, а также эффективности формирования программы компилятором. Этими тестами служат наборы SPECintXX и SPECfpXX, где символами XX обозначен год разработки теста; время на операции ввода-вывода ничтожно мало. Тест SPECintXX состоит из нескольких программ для различных прикладных областей (теория цепей, разработка логических схем, упаковка текстовых файлов, электронные таблицы и т. п.). Программы написаны на языке Си. Тест определяет производительность процессора при работе с целыми числами. Тест SPECfpXX служит для оценки производительности при наличии операций с ПТ.

Кроме этих часто обновляемых тестов (после названия указывается год) существует и ряд других, в частности для оценки производительности многопроцессорных систем (SPECrate), оценки машин при обработке трансакций (TPC), рабочих станций, серверов (AIM) и т. п. Эти тесты стремятся учесть влияние, оказываемое на производительность машины различными ее компонентами, а не только процессором.

Компьютер принято характеризовать следующими технико-эксплуатационными параметрами:

быстродействие или производительность;

надежность;

разрядность обрабатываемых слов и шин интерфейсов;
емкость оперативной памяти;
емкость накопителя на жестком магнитном диске (НЖМД);
емкость кэш-памяти;
стоимость;
размеры и масса.

Однако этих параметров может быть недостаточно, чтобы узнать все возможности компьютера. Если компьютер предполагается использовать в сети, то нужно знать о наличии и типе сетевых средств, типе установленной ОС. Для персонального компьютера важным является тип монитора и видеоадаптера, а для компьютера общего назначения — возможные режимы обслуживания пользователей, наличие средств для работы нескольких пользователей, число одновременно выполняемых программ, наличие специальных средств защиты памяти и т. п.

1.5. Вычислительные системы и сети

Для повышения надежности и производительности начиная с 1960-х гг. несколько компьютеров связывались между собой, образуя многомашинные и многопроцессорные вычислительные системы и комплексы. Вначале такие комплексы предназначались для военных целей. Связь между отдельными компьютерами в них осуществлялась за счет доступа к общим наборам данных с помощью совместно используемых ВЗУ, т. е. накопителей на магнитных дисках (НМД) и лентах (НМЛ). Позднее для ускорения взаимодействия компьютеров стали использовать специальные адаптеры.

Если в вычислительной системе предусматривается несколько процессоров и они имеют доступ к общим данным, находящимся в оперативной памяти, а также могут взаимодействовать со всеми периферийными устройствами, то такой комплекс принято называть *многопроцессорным*. Многопроцессорные комплексы обладают большей надежностью и гибкостью, но для их создания требуются специальные дорогостоящие средства коммутации.

Часто компьютеры расположены на значительных расстояниях друг от друга, а для их работы могут требоваться данные из компьютера, находящегося в другом городе или стране. Тогда для их объединения между собой используют сети передачи данных. Несколько компьютеров, объединенных сетями передачи данных, называются *вычислительной сетью*. Если компьютеры удалены друг от друга и для их объединения использованы стандартные телефонные каналы — это *глобальная вычислительная сеть* (ГВС).

Появление мини- и микроЭВМ, а несколько позже и персональных компьютеров, призванных обслуживать небольшие группы или даже отдельных пользователей, привело к тому, что полу-

ченные результаты обработки в отдельных компьютерах предназначались для совместного использования. Поэтому был разработан способ объединения этих компьютеров в *локальную вычислительную сеть* (ЛВС) — совокупность компьютеров (обычно персональных), расположенных на незначительных расстояниях друг от друга (на одном этаже, в одном здании или нескольких близлежащих зданиях) и соединенных между собой высокоскоростными каналами связи. Кроме того, для работы сети необходимо специальное ПО, обеспечивающее взаимодействие между процессами в отдельных компьютерах. К концу 1980-х гг. утвердились стандартные технологии объединения компьютеров в локальные сети. Появились два типа сетей — одноранговые (peer-to-peer) и построенные по типу «клиент-сервер», а в качестве рабочих станций в них стали использоваться получавшие все более широкое признание персональные компьютеры. Со временем были разработаны стандартные сетевые технологии, превратившие процесс создания локальной сети в достаточно простую работу. Благодаря новому коммуникационному оборудованию — коммутаторам, маршрутизаторам, шлюзам — и новым каналам связи стало возможным объединять в сеть тысячи компьютеров. Для управления такими корпоративными сетями требовались мощные серверы, в качестве которых стали использовать мейнфреймы.

Контрольные вопросы

1. Что представляет собой информация, хранящаяся в оперативной памяти?
2. Что понимают под сообщением?
3. Что служит алфавитом компьютера? Какие символы образуют алфавит?
4. Дайте определение биту. Что такое байт? Сколько бит он содержит?
5. Какова длина машинного слова современного компьютера? Какова длина команды в персональном компьютере?
6. Опишите состав процессора персонального компьютера.
7. Какие устройства входят в состав памяти машины? Что такое ПЗУ, ОЗУ, ВЗУ?
8. Что представляет собой BIOS и где находится эта программа?
9. Как производится начальная загрузка персонального компьютера?
10. Опишите действия, производимые в персональном компьютере при его включении.
11. Расскажите о назначении триггера, счетчика, регистра.
12. Каков состав ПО компьютера?
13. Как можно определить производительность и надежность компьютера?
14. Какие существуют методы повышения надежности компьютера?
15. Как добиться объединения ресурсов отдельных компьютеров? С какой целью производится такое объединение?
16. Когда и с какой целью были разработаны ЛВС?

ГЛАВА 2

ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В КОМПЬЮТЕРЕ

2.1. Системы счисления

В компьютере может храниться и обрабатываться информация различного характера: числа, адреса, команды, различные символы, графические изображения и т.д. Любая информация в компьютере представляется в числовой форме, при этом используются различные системы счисления.

Под *системой счисления* понимается способ представления чисел с помощью символов (цифр), имеющих определенное количественное значение. В любой системе счисления числа представляются в виде последовательности цифр. Каждая цифра числа занимает определенное место, называемое *позицией* или *разрядом*. Разряды нумеруются, обычно слева направо начиная с нулевого разряда. Количество различных цифр, используемых в позиционной системе счисления, называется ее *основанием*. Таким образом, число A с основанием q, состоящее из n разрядов, записывается в следующем виде:

$$A_q = a_{n-1}a_{n-2} \ldots a_i \ldots a_2a_1a_0,$$

где a_i — цифры числа A (i = 0, 1, ..., n – 1).

Существуют позиционные и непозиционные системы счисления. В позиционных системах значение цифры зависит от ее позиции. Например в десятичном числе A_{10} = 121 левая цифра «1» имеет значение «сто», цифра «2» — значение «двадцать», а правая цифра «1» — значение «один».

В зависимости от величины основания различают двоичную, восьмеричную, десятичную, шестнадцатеричную и другие системы счисления. В двоичной системе счисления используют две цифры (0 и 1), в восьмеричной — восемь (0, 1, ..., 7), в десятичной — десять (0,1, ..., 9).

В шестнадцатеричной системе счисления необходимо использовать 16 цифр. Чтобы не изображать одной цифрой два символа, кроме десятичных цифр применяют шесть первых прописных букв латинского алфавита, имеющих следующие значения: A = 10, B = 11, C = 12, D = 13, E = 14, F = 15. В любой системе счисления

младшая цифра равна нулю, а старшая на единицу меньше основания.

В позиционных системах счисления каждая цифра имеет вес. Обычно вес старшей цифры по отношению к весу соседней младшей цифры больше в количество раз, равное основанию системы счисления. При этом для целых чисел вес младшего разряда в любой системе счисления равен единице. Тогда значение целого числа в системе счисления с основанием q может быть определено следующим образом:

$$A_q = (a_{n-1}a_{n-2} \ldots a_2a_1a_0)_q = a_{n-1}q^{n-1} + a_{n-2}q^{n-2} + \ldots + a_2q^2 + a_1q^1 + a_0q^0.$$

В общем случае число состоит из целой и дробной частей. Тогда его значение определяется из следующего соотношения:

$$A_q = (a_{n-1}a_{n-2} \ldots a_2a_1a_0a_{-1}a_{-2} \ldots a_{-m})_q =$$

$$= a_{n-1}q^{n-1} + a_{n-2}q^{n-2} + \ldots + a_2q^2 + a_1q^1 + a_0q^0 + a_{-1}q^{-1} +$$

$$+ a_{-2}q^{-2} + \ldots + a_{-m}q^{-m},$$

где n — число разрядов целой части числа; m — число разрядов дробной части числа.

В качестве примера приведем запись десятичного числа $A_{10} = 46{,}25$ в системах счисления с основаниями 2, 8 и 16:

$$A_2 = (101110{,}01)_2 = (1 \cdot 2^5 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^{-2})_{10} =$$
$$= (32 + 8 + 4 + 2 + {}^1\!/_4)_{10} = 46{,}25_{10};$$

$$A_8 = (56{,}2)_8 = (5 \cdot 8^1 + 6 \cdot 8^0 + 2 \cdot 8^{-1})_{10} = (40 + 6 + {}^2\!/_8) = 46{,}25_{10};$$

$$A_{16} = (2\text{E},4)_{16} = (2 \cdot 16^1 + 14 \cdot 16^0 + 4 \cdot 16^{-1})_{10} =$$
$$= (32 + 14 + {}^4\!/_{16})_{10} = 46{,}25_{10}.$$

При хранении и обработке информации внутри компьютера используется двоичная система счисления. Это объясняется необходимостью физического представления только двух цифр (0 и 1), простотой выполнения арифметических операций и возможностью осуществления любых преобразований информации с помощью двоичных логических элементов.

Шестнадцатеричная (и реже восьмеричная) система счисления используется для более компактного представления информации (по сравнению с двоичной системой) при вводе и выводе больших массивов двоичных данных. Это связано с простотой перехода от двоичной системы счисления к шестнадцатеричной (восьмеричной) и наоборот.

Для перевода числа из двоичной в шестнадцатеричную (восьмеричную) систему счисления следует разделить двоичное число на группы по четыре (три) разряда, начиная от запятой влево и впра-

во, а недостающие до полной группы разряды заполнить нулями. Затем заменить каждую группу шестнадцатеричной (восьмеричной) цифрой. Например:

$$A_2 = (101110{,}01)_2 = (0010\ 1110{,}0100)_2 = (2E{,}4)_{16};$$

$$A_2 = (101110{,}01)_2 = (101\ 110{,}010)_2 = (56{,}2)_8.$$

При переводе из шестнадцатеричной (восьмеричной) системы счисления в двоичную достаточно заменить каждую шестнадцатеричную (восьмеричную) цифру соответствующим четырехразрядным (трехразрядным) двоичным числом. Например:

$$A_{16} = (2B7{,}5)_{16} = (0010\ 1011\ 0111{,}0101)_2 = (1010110111{,}0101)_2;$$

$$A_8 = (1042{,}3)_8 = (001\ 000\ 100\ 010{,}011)_2 = (1000100010{,}011)_2.$$

Существуют универсальные алгоритмы перевода чисел из одной системы счисления в другую, при этом перевод целых и дробных чисел выполняется различным образом.

Рассмотренные выше системы счисления являются позиционными. Примером непозиционной системы является римская система счисления. В ней значение цифры не зависит от ее позиции. Например, десятичное число 32 представляется в римской системе счисления в виде числа XXXII, в котором все цифры X имеют значение «десять», а все цифры I — значение «один».

Исключением являются числа IV и VI, в которых цифра I принимает значение «минус один» и «плюс один» соответственно. Недостатком римской системы счисления является сложность представления больших чисел, а также выполнения арифметических операций.

Существуют также позиционные системы счисления, использующие одновременно несколько оснований. Примером является двоично-десятичная система счисления. Она используется как вспомогательная при вводе и выводе данных, а также в компьютерах с десятичной арифметикой. Для записи десятичного числа в двоично-десятичной системе счисления каждая десятичная цифра заменяется соответствующим четырехразрядным двоичным числом (тетрадой). Например:

$$A_{10} = (1951)_{10} = (0001\ 1001\ 0101\ 0001)_{2\text{–}10}.$$

В двоично-десятичной системе счисления для представления десятичных цифр используется двоичная система, в то же время веса десятичных цифр (двоичных тетрад) остаются теми же, что и в десятичной системе. Так, в рассмотренном выше примере старшая тетрада 0001 имеет значение «одна тысяча», младшая — значение «один».

Для представления десятичных цифр может использоваться как обычная двоичная система с весами разрядов 8421 (код прямого замещения), так и двоичные системы с другими весами, например 2421. Использование таких весов позволяет упростить выполнение арифметических операций непосредственно над двоично-десятичными кодами без перевода их в двоичную систему счисле-

ния. Это может быть целесообразно при несложной обработке больших массивов десятичных чисел.

В двоичной тетраде может быть записано $2^4 = 16$ различных комбинаций, из которых только 10 комбинаций используются для кодирования десятичных цифр, остальные шесть являются запрещенными. Таким образом, существует возможность различного кодирования десятичных цифр. Количество всех возможных вариантов кодирования равно числу сочетаний по 10 из 16. При выборе вида двоично-десятичного кодирования следует учитывать следующие требования:

1. Различные десятичные цифры должны кодироваться различными двоичными тетрадами.

Таблица 2.1

Двоично-десятичные коды

Десятичные цифры	Двоичные тетрады Д-кодов				
	Обозначение				
	Д1	Д2	Д3	Д4	Д5
	Веса разрядов тетрады				
	8421	2421	5121	8421 + 3	5321
0	0000	0000	0000	0011	0000
1	0001	0001	0001	0100	0001
2	0010	0010	0010	0101	0010
3	0011	0011	0011	0110	0011
4	0100	0100	0111	0111	0101
5	0101	1011	1000	1000	1000
6	0110	1100	1100	1001	1001
7	0111	1101	1101	1010	1010
8	1000	1110	1110	1011	1100
9	1001	1111	1111	1100	1101
Запрещенные комбинации	1010	0101	0100	0000	0100
	1011	0110	0101	0001	0110
	1100	0111	0110	0010	0111
	1101	1000	1001	1101	1011
	1110	1001	1010	1110	1110
	1111	1010	1011	1111	1111

2. Большей по значению десятичной цифре должна соответствовать бóльшая по значению двоичная тетрада.

3. Двоично-десятичный код должен обладать свойством самодополняемости. Это свойство заключается в том, что двум тетрадам, которые получаются друг из друга инвертированием, должны соответствовать десятичные цифры, дополняющие друг друга до девяти. Свойство самодополняемости позволяет получать обратный код тетрады простым инвертированием (см. подразд. 2.3).

4. Желательно, чтобы двоичные разряды тетрады имели определенные веса.

В качестве примера в табл. 2.1 приведены некоторые из возможных двоично-десятичных кодов (Д-коды).

Код Д1 является кодом прямого замещения; он не обладает свойством самодополняемости. Коды Д2, Д3 и Д4 являются самодополняющимися. Код Д4 представляет собой код, тетрады которого получаются из тетрад кода Д1 их увеличением на $3_{10} = 0011_2$. Код Д5 не является самодополняющимся.

2.2. Формы представления чисел

В общем случае числа имеют знак (плюс или минус). Кроме того, число может включать в себя целую и дробную части. Специальные формы представления чисел позволяют кодировать знаки чисел и указывать положение точки (запятой), разделяющей целую и дробную части числа.

Для кодирования знака числа отводится специальный разряд, называемый *знаковым*. Под него обычно отводится старший разряд числа. Для положительных чисел в нем записывается цифра 0, для отрицательных — 1.

Для указания положения точки используют одну из двух форм: форму с фиксированной или форму с плавающей точкой.

Форма представления чисел с фиксированной точкой. Эта форма, называемая также естественной, предполагает, что все числа в компьютере могут быть только целыми или только дробными. В этом случае положение точки является стандартным для данного компьютера и не требует специального указания. Эта форма является простой, но приводит к некоторому усложнению программирования.

Если в компьютере для всех чисел положение точки зафиксировано справа от младшего цифрового разряда, то числа принимают только целые значения.

На рис. 2.1 представлено $(n + 1)$-разрядное целое число. Один разряд занимает знак, остальные n разрядов используются как цифровые. Веса цифровых разрядов показаны в верхней части ри-

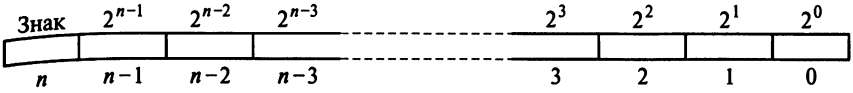


Рис. 2.1. Целое число в форме с фиксированной точкой

сунка. В этом случае в компьютере могут быть представлены числа, модуль которых находится в диапазоне

$$2^0 \le |A| < 2^{n-1}.$$

При этом точность представления чисел равна единице, так как числа могут быть только целыми.

Если точка зафиксирована слева от старшего цифрового разряда, то все числа могут быть только дробными. Формат дробного числа с фиксированной точкой (ФТ) представлен на рис. 2.2.

Дробные числа с ФТ, имеющие n цифровых разрядов, представляются с точностью 2^{-n} (величина единицы младшего разряда дроби) в диапазоне

$$2^{-n} \le |A| \le 1 - 2^{-n}.$$

Напомним, что ФТ никак не представляется в компьютере, ее положение определяется однозначно принятой для данного компьютера формой чисел (целые или дробные).

В формате с ФТ могут представляться числа без знака. В этом случае все разряды являются цифровыми.

В современных микропроцессорах используется представление данных в форме целых чисел с ФТ. Форма дробных чисел с ФТ применяется для представления мантиссы числа в форме с ПТ. Некоторые из целочисленных форматов микропроцессоров фирмы Intel представлены на рис. 2.3.

Стандартными форматами являются байт, слово и двойное слово.

При обработке мультимедийной информации используются не только отдельные целые числа, но и группы целых чисел, которые обрабатываются одновременно. При этом несколько малоразрядных чисел упаковываются в 64-разрядное слово. Упакованными могут быть восемь байтов, четыре слова или два двойных слова.

Форма представления чисел с плавающей точкой. Эта форма (нормальная или полулогарифмическая) позволяет представлять

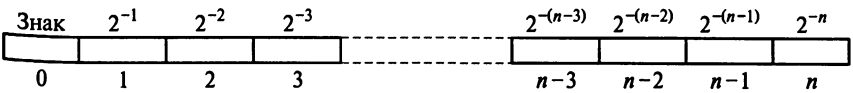


Рис. 2.2. Дробное число в форме с фиксированной точкой

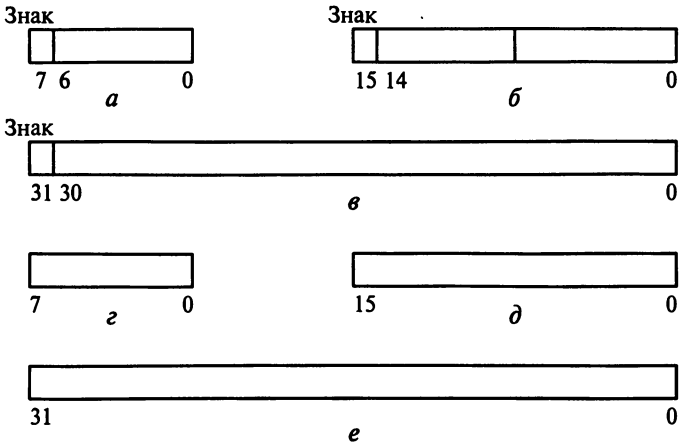


Рис. 2.3. Целочисленные форматы микропроцессоров фирмы Intel:

а — байт (целое со знаком); *б* — слово (целое со знаком); *в* — двойное слово (целое со знаком); *г* — байт (целое без знака); *д* — слово (целое без знака); *е* — двойное слово (целое без знака)

в компьютере любые (целые, дробные или смешанные) числа. Число в форме с ПТ записывается в виде двух частей: мантиссы и порядка. Мантисса включает в себя значащие разряды числа, а порядок указывает положение точки. При этом мантисса записывается как дробное число с ФТ, а порядок — как целое число с ФТ.

Знак мантиссы является знаком всего числа, а знак порядка определяет, содержит ли число целую часть.

Значение числа с ПТ определяется следующим образом:

$$A_q = mq^p,$$

где m — мантисса числа; q — основание системы счисления; p — порядок числа.

Так, в десятичной системе счисления число $A_{10} = -123{,}456$ в форме с ПТ может быть записано следующим образом:

$$m_A = -0{,}123456,\ p_A = +3;$$
$$m_A = -0{,}0123456,\ p_A = +4;$$
$$m_A = -0{,}00123456,\ p_A = +5 \text{ и т.д.}$$

При заданных значениях мантиссы и порядка для определения значения числа нужно точку (запятую) в мантиссе перенести на количество разрядов, равное величине порядка, вправо для положительных порядков и влево для отрицательных. Например:

$$m_A = -0{,}87412456,\ p_A = +5 \rightarrow A = -87412{,}456;$$
$$m_B = +0{,}12437696,\ p_B = -2 \rightarrow B = +\ 0{,}0012437696.$$

Число с ПТ может быть нормализованным и ненормализованным. Число не нормализовано, если старшая цифра мантиссы равна нулю, т.е. $|m| < 1/q$. Нормализованное число с ПТ позволяет сохранить большее количество значащих цифр, поэтому в памяти числа хранятся в нормализованном виде. Нормализация выполняется путем сдвига мантиссы влево до тех пор, пока старший разряд мантиссы не станет равным единице. Так как значение мантиссы при этом увеличивается, для сохранения величины числа при сдвиге на каждый разряд значение порядка уменьшается на единицу.

Нормализованное число с ПТ представляется с точностью 2^{-n}, где n — разрядность мантиссы. Диапазон чисел с ПТ составляет:

$$\frac{1}{2}\cdot 2^{-(2^k-1)} \le |A| \le \left(1-2^{-n}\right)2^{(2^k-1)},$$

где k — разрядность порядка.

Для увеличения диапазона чисел с ПТ (за счет некоторого уменьшения точности) двоичная мантисса может рассматриваться как шестнадцатеричное число. В этом случае каждая двоичная тетрада представляет одну шестнадцатеричную цифру, поэтому нормализация будет нарушена лишь тогда, когда четыре старших разряда мантиссы будут равны нулю. Например, если старшие разряды мантиссы имеют вид 0, 000101..., то мантисса считается нормализованной, так как тетрада 0001 представляет шестнадцатеричную цифру 1. Появление незначащих нулей в мантиссе приводит к потере точности. Вместе с тем существенно увеличивается диапазон представления чисел:

$$\frac{1}{16}\cdot 16^{-(2^{-k}-1)} \le |A| \le \left(1-16^{-\frac{m}{4}}\right)16^{(2^k-1)}.$$

Порядок числа может быть положительным или отрицательным. Для упрощения операций над порядками часто используют смещенный порядок путем увеличения действительного порядка на величину 2^k, где k — число разрядов порядка. При этом смещенный порядок всегда является положительным числом и поэтому его знак не указывается. Пример формата n-разрядного числа с ПТ и смещенным порядком представлен на рис. 2.4. Для знака числа отводится старший разряд, k разрядов занимает смещенный

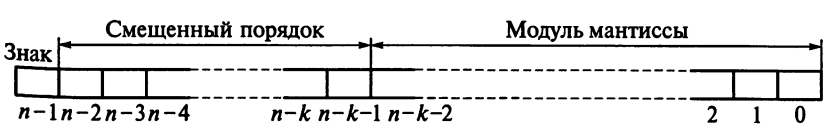


Рис. 2.4. Формат числа с плавающей точкой

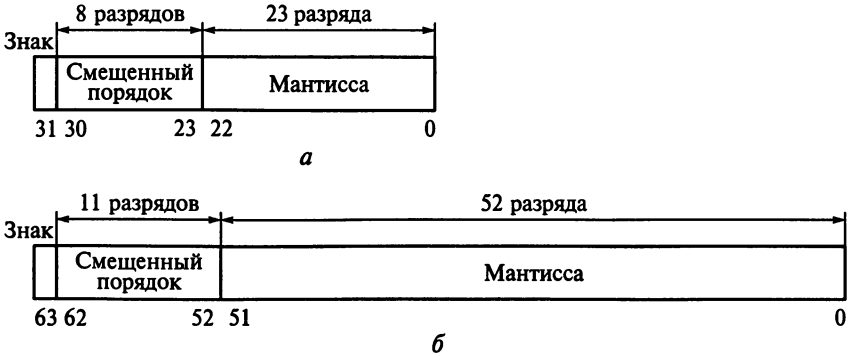


Рис. 2.5. Основные форматы чисел с плавающей точкой:
а — одинарный; *б* — двойной

порядок, а остальные $n - k - 1$ выделяются под мантиссу. Диапазон и точность представляемых в форме с ПТ чисел зависят от формата.

Рекомендуемые стандартом основные форматы чисел с ПТ представлены на рис. 2.5. Одинарный формат занимает 32 разряда, двойной — 64. Обычно для повышения точности используют способ скрытой единицы.

Суть способа заключается в том, что в нормализованном числе старший разряд мантиссы всегда равен единице, поэтому его можно не записывать, а подразумевать. Освободившийся разряд используется для записи дополнительного разряда мантиссы. Перед выполнением арифметических операций подразумеваемый разряд восстанавливается.

В одинарном формате под смещенный порядок отводится восемь разрядов, и под мантиссу — 24 (с учетом скрытой единицы). При этом диапазон представления чисел составляет: $10^{-38} \le |A| \le 10^{+38}$. В двойном формате смещенный порядок занимает 11 раз-

Байт		Байт		...	Байт		Байт	
Зона	Цифра	Зона	Цифра		Зона	Цифра	Знак	Цифра

а

Байт		Байт		Байт		Байт		Байт	
Зона	1	Зона	2	Зона	9	Зона	8	Минус	7
1111	0001	1111	0010	1111	1001	1111	1000	1101	0111

б

Рис. 2.6. Зонный формат двоично-десятичных чисел:
а — структура формата; *б* — пример записи числа –12 987

Байт		Байт		...	Байт		Байт	
Цифра	Цифра	Цифра	Цифра		Цифра	Цифра	Цифра	Знак

а

Байт		Байт		Байт		Байт	
0	2	3	7	4	6	5	Плюс
0000	0010	0011	0111	0100	0110	0101	1100

б

Рис. 2.7. Упакованный формат двоично-десятичных чисел:
а — структура формата; *б* — пример записи числа +237 465.

рядов, мантисса — 53, а диапазон представления чисел составляет: $10^{-308} \le |A| \le 10^{+308}$.

Как и целые числа, числа с ПТ могут быть записаны в упакованном формате. В микропроцессорах фирмы Intel четыре числа одинарной точности (по 32 разряда) упаковываются в группу длиной 128 разрядов. В такую же группу упаковываются два числа двойной точности (по 64 разряда).

Форматы двоично-десятичных чисел. Такие числа могут быть представлены в двух форматах: зонном и упакованном. В этих форматах каждая десятичная цифра и знак числа заменяются двоичной тетрадой в соответствии с используемым двоично-десятичным кодом. Количество цифр в числе может быть произвольным.

В *зонном* формате под каждую десятичную цифру отводится байт (рис. 2.6). В старшей тетраде записывается код зоны (например, код 1111), а в младшей — код цифры. В младшем байте вместо кода зоны записывается знак числа.

В *упакованном* формате байт содержит две цифры (рис. 2.7). Младшая тетрада последнего байта содержит знак числа.

2.3. Машинные коды

Для упрощения арифметических операций числа записываются в специальной форме, называемой *машинными кодами*. Эти коды позволяют:

свести операцию вычитания к операции сложения;
автоматически получать знак суммы (разности);
выявлять переполнение разрядной сетки.

Существуют следующие коды:

прямой (ПК);
обратный (ОК);
дополнительный (ДК).

Положительные числа во всех кодах записываются одинаково. Наиболее просто числа записываются в ПК, для этого достаточно в знаковом разряде записать знак («0» или «1»), а цифровые разряды оставить без изменения. Например, для чисел с ФТ прямой код чисел получается следующим образом:

$$A_2 = +1001010 \rightarrow \text{А}^{\text{ПК}} = 0\ 1001010;$$

$$A_2 = -1001010 \rightarrow \text{А}^{\text{ПК}} = 1\ 1001010.$$

Прямой код числа A определяется следующим соотношением:

$$A^{\text{ПК}} = A \text{ при } A \geq 0;$$

$$A^{\text{ПК}} = Z + |A| \text{ при } A < 0,$$

где Z — вес знакового разряда.

При сложении чисел с разными знаками фактически необходимо выполнять вычитание, а эта операция является менее удобной, чем сложение. Обратный и дополнительный коды позволяют свести операцию вычитания к операции сложения. В ПК этого сделать нельзя.

Обратный код отрицательного числа получают из ПК инвертированием цифровых разрядов. Положительные числа в ОК записываются так же, как и в прямом. Например:

$$A_2 = +1011011 \rightarrow A^{\text{ПК}} = 0\ 1011011 \rightarrow A^{\text{ОК}} = 0\ 1011011;$$

$$A_2 = -1011011 \rightarrow A^{\text{ПК}} = 1\ 1011011 \rightarrow A^{\text{ОК}} = 1\ 0100100.$$

Обратный код числа A определяется следующим образом:
если A — дробное число с ФТ,

$$A^{\text{ОК}} = 2 - 2^{-(n-1)} - |A|; \qquad (2.1)$$

если A — целое число с ФТ,

$$A^{\text{ОК}} = 2^n - 1 - |A|,$$

где n — разрядность числа.

Особенностью ОК является различное представление цифровых разрядов положительного и отрицательного нулей:

$$A_2 = +0000000 \rightarrow A^{\text{ПК}} = 0\ 0000000 \rightarrow A^{\text{ОК}} = 0\ 0000000;$$

$$A_2 = -0000000 \rightarrow A^{\text{ПК}} = 1\ 0000000 \rightarrow A^{\text{ОК}} = 1\ 1111111.$$

Для получения ПК отрицательного числа из ОК цифровые разряды инвертируют:

$$A^{\text{ОК}} = 1\ 0011101 \rightarrow A^{\text{ПК}} = 1\ 1100010 \rightarrow A_2 = -1100010.$$

Дополнительный код отрицательного числа получают из ОК прибавлением единицы к младшему разряду. Положительные числа в ДК записываются так же, как и в прямом. Например:

$$A_2 = +1110101 \rightarrow A^{ПК} = 0\ 1110101 \rightarrow A^{ОК} = \\ = 0\ 1110101 \rightarrow A^{ДК} = 0\ 1110101;$$

$$A_2 = -1110101 \rightarrow A^{ПК} = 1\ 1110101 \rightarrow A^{ОК} = \\ = 1\ 0001010 \rightarrow A^{ДК} = 1\ 0001011.$$

Дополнительный код числа A определяется из формулы (2.1) с учетом прибавления единицы младшего разряда:

$$A^{ОК} = 2 - |A|, \text{ если } A \text{ — дробное число с ФТ;}$$

$$A^{ОК} = 2^n - |A|, \text{ если } A \text{ — целое число с ФТ.}$$

Нуль в ДК имеет единственное представление:

$$A_2 = +0000000 \rightarrow A^{ПК} = 0\ 0000000 \rightarrow A^{ОК} = \\ = 0\ 0000000 \rightarrow A^{ДК} = 0\ 0000000;$$

$$A_2 = -0000000 \rightarrow A^{ПК} = 1\ 0000000 \rightarrow A^{ОК} = \\ = 1\ 1111111 \rightarrow A^{ДК} = 0\ 0000000.$$

Для получения ПК отрицательного числа из ДК сначала инвертируются цифровые разряды, а затем к младшему разряду прибавляется единица:

$$A^{ДК} = 1\ 0001011 \rightarrow 1\ 1110100 \rightarrow A^{ПК} = 1\ 1110101 \rightarrow A_2 = -1110101.$$

Обратный и дополнительный коды с точки зрения простоты выполнения операций над числами равноценны. Однако операции с удвоенной разрядностью в ДК выполняются проще, поэтому его чаще используют.

2.4. Кодирование текстовой информации

Кроме числовой информации в компьютере может обрабатываться и текстовая информация, содержащая буквы, цифры, знаки препинания и другие символы. Обычно число различных символов не превышает 256, поэтому для представления символов в компьютере используют восьмиразрядные двоичные коды (байты). Существуют различные системы кодирования текстовой информации — коды КОИ8, ДКОИ, ASCII. В компьютерах IBM PC используется код ASCII. Код символа в нем занимает восемь разрядов. Стандарт IOSO предусматривает несколько модификаций кода ASCII в зависимости от национального алфавита. Кодирование символов в одной из модификаций кода ASCII показано в табл. 2.2.

В настоящее время получает развитие 16-разрядный код Unicode, который позволяет одновременно закодировать все буквы всех известных языков. Для букв русского языка в нем предусмотрены коды 1040...1093. Впервые Unicode использовался в Windows NT.

Таблица 2.2

Кодирование символов в коде ASCII

Десятичное значение ➡	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
⬇ Шестнадцатеричное значение	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	Пустой символ (0)	►	Пробел	0	@	P	‹	p	Ç	È	á	¼	└	╨	∝	≡
1	1	☺	◄	!	1	A	Q	a	q	ü	Æ	í	½	┴	╤	β	±
2	2	☻	↕	"	2	B	R	b	r	é	FE	ó	¾	┬	╥	γ	≥
3	3	♥	‼	#	3	C	S	c	s	â	ô	ú	|	├	╙	π	≤
4	4	♦	¶	$	4	D	T	d	t	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	♣	§	%	5	E	U	e	u	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	♠	▬	&	6	F	V	f	v	å	û	ª	╢	╞	╓	μ	÷
7	7	Гудок •	↨	'	7	G	W	g	w	ç	ù	º	╖	╟	╫	τ	≈
8	8	•	↑	(	8	H	X	h	x	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	○	↓	)	9	I	Y	i	y	ë	Ö	⌐	╣	╔	┘	Θ	•
10	A	◙	→	*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Ω	·
11	B	♂	ESC ←	+	;	K	[	k	{	ï	¢	½	╗	╦	■	δ	√
12	C	♀	∟	,	<	L	\	l	¦	î	£	¼	╝	╠		∞	n
13	D	♪	↔	—	=	M	]	m	}	ì	¥	¡	╜	═		∅	²
14	E	♫	▲	·	>	N	^	n	~	Ä	P_t	«	╛	╬		∈	■
15	F	☼	▼	/	?	O	_	o	⌂	Å	f	»	┐	╧		∩	Пустой символ (FF)

Контрольные вопросы

1. Что такое система счисления?
2. Что такое основание системы счисления?
3. Как зависит разрядность чисел от величины основания системы счисления?
4. Почему информация в компьютере представляется в двоичной системе счисления?
5. Какие системы счисления используются для представления информации в компьютере?
6. Для чего применяются двоично-десятичные коды?
7. Чем отличаются формы представления чисел с фиксированной и плавающей точками?
8. Как кодируются знаки чисел?
9. Можно ли различить форматы целых и дробных чисел с ФТ?

10. Какие элементы формата чисел с ПТ вы знаете?

11. Чем отличается нормализованное число с ПТ от ненормализованного числа?

12. Для чего применяются смещенные порядки?

13. Какой прием используется для расширения диапазона чисел с ПТ?

14. В чем заключается способ скрытой единицы?

15. Охарактеризуйте форматы двоично-десятичных чисел.

16. Как кодируются знаки двоично-десятичных чисел?

17. Для чего используются машинные коды чисел?

18. Чем отличаются дополнительный и обратный коды отрицательных чисел?

19. Как из ДК отрицательного числа получить ПК числа?

20. Каким образом кодируется текстовая информация?

ГЛАВА 3

ЭЛЕМЕНТЫ И ТИПОВЫЕ УЗЛЫ КОМПЬЮТЕРА

3.1. Составные части компьютера

Современные компьютеры представляют собой технические системы, отличающиеся сложной структурой, большим числом электронных элементов и электромеханических деталей, а также сложностью выполняемых функций. Поэтому изучение компьютера целесообразно начать с рассмотрения простейших частей, из которых он состоит. При множестве электронных элементов (миллионы), используемых в компьютере, число их типов сравнительно невелико (десятки и сотни). Дело в том, что все устройства (например, оперативная память) имеют регулярную структуру, т.е. состоят из большого числа повторяющихся (типовых) схем.

Преобразование информации в компьютере выполняется при помощи электронных схем, имеющих различную сложность. По функциональной сложности принято делить электронные схемы компьютера на элементы, узлы (блоки) и устройства.

Элемент — это простейшая часть компьютера, выполняющая операции над двоичными цифрами (битами). Основные элементы могут быть логическими или элементами памяти. Логические элементы выполняют двоичные (бинарные) операции, на основе которых осуществляются практически все преобразования информации. В качестве логических элементов используются элементы И, ИЛИ, НЕ, И—НЕ, ИЛИ—НЕ, И—ИЛИ—НЕ и т.д. Элементы памяти чаще всего представляют собой триггеры различных типов: *RS*-, *JK*-, *D*- или *T*-триггеры. Кроме логических элементов и элементов памяти в состав компьютера входят вспомогательные элементы, усиливающие или формирующие сигналы стандартной формы.

Узлы (блоки) состоят из элементов и выполняют операции над байтами или словами, состоящими из нескольких байтов. К типовым узлам компьютера относятся регистры, счетчики, сумматоры, дешифраторы, селекторы, мультиплексоры и др. Несколько узлов могут объединяться в функциональные блоки (например, блок сумматора может включать в себя собственно сумматор и регистр сумматора).

Устройства компьютера строятся из элементов и узлов и выполняют определенный набор однотипных операций. К устройствам относятся запоминающие устройства, арифметико-логическое устройство, центральное устройство управления, устройства ввода и вывода. Устройства компьютера конструктивно выполняют отдельно или несколько устройств объединяют в один конструктивный блок (например, системный блок может объединять устройство управления, арифметико-логическое устройство и устройства памяти).

В зависимости от состава узлы могут быть комбинационного (комбинационные схемы — КС) или накапливающего типа (автоматы с памятью, последовательные схемы).

Узлы комбинационного типа состоят из логических элементов. Их главная особенность заключается в том, что выходной сигнал (Y) зависит только от комбинации входных сигналов (X) в данный момент времени, при этом каждой комбинации сигналов на входе соответствует выходной сигнал (рис. 3.1, *а*). Выходной сигнал может измениться только при получении другого входного сигнала. При неоднократном повторении одного и того же входного сигнала значение выходного также повторяется, поэтому комбинационная схема фактически осуществляет перекодировку входных сигналов в выходные. Несмотря на кажущуюся примитивность логики работы комбинационных схем, все основные преобразования информации в компьютере выполняются с их помощью. Это объясняется тем, что в основе преобразования данных в компьютере заложено выполнение логических операций.

Автоматы с памятью состоят из логических элементов и элементов памяти (рис. 3.1, *б*). Информация, записанная в памяти автомата, называется *состоянием автомата* (Q). Выходной сигнал автомата в общем случае зависит от сигнала на входе и состояния автомата, поэтому при одном и том же входном сигнале автомат

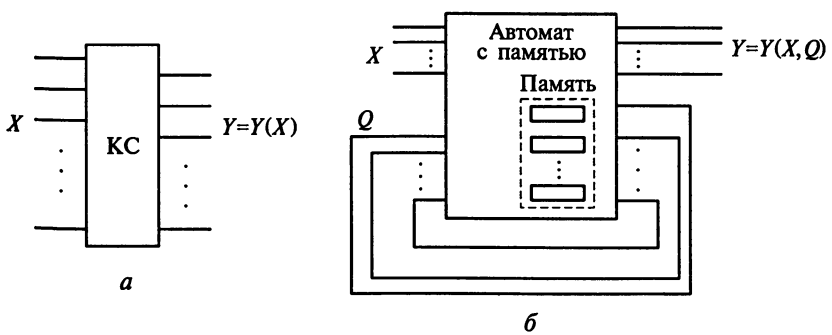


Рис. 3.1. Комбинационные схемы и автоматы с памятью:
а — комбинационная схема; *б* — автомат с памятью

может выдавать различные выходные сигналы. При работе автомата в его памяти накапливается обобщенная информация о всех входных сигналах, поступивших к данному моменту времени, поэтому состояние автомата и выходной сигнал зависят от всей предыстории входных сигналов. Наличие памяти позволяет автомату выполнять не только отдельные операции, но и последовательности взаимосвязанных операций, т.е. заданные алгоритмы обработки данных. Компьютер в целом представляет собой сложный автомат с памятью большой емкости. Для понимания организации компьютера и процессов преобразования информации в нем необходимо изучить логику работы его составных частей, прежде всего состав и работу элементов и типовых узлов.

3.2. Логические элементы

Логические элементы выполняют преобразования информации, которые описываются логическими функциями. Логическая функция и ее аргументы могут принимать только два значения: «0» и «1». Зависимость значений логической функции от значений ее аргументов можно задать с помощью таблицы истинности (табл. 3.1).

Логическая функция И (логическое умножение, конъюнкция) $F(x_1, x_2) = x_1 x_2$ принимает значение «1», если оба аргумента равны «1», а функция ИЛИ (логическое сложение, дизъюнкция) $F(x_1, x_2) = x_1 \vee x_2$ — если один или несколько аргументов равны «1».

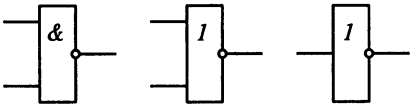
Функция НЕ (инверсия, отрицание) принимает значение «1», если аргумент равен «0» и наоборот. Таким образом, значения функций И, ИЛИ и НЕ определяются в соответствии с обычным смыслом слов «и», «или» и «не». Число входов элементов ИЛИ и И совпадает с числом аргументов соответствующей функции. Условные графические обозначения (УГО) элементов И, ИЛИ и НЕ на функциональных схемах приведены на рис. 3.2.

Таблица 3.1

Таблица истинности логических функций двух аргументов

Аргументы		Функции							
x_1	x_2	x_1x_2	$x_1 \vee x_2$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_1x_2}$	$\overline{x_1 \vee x_2}$	$x_1 \sim x_2$	$x_1 \oplus x_2$
0	0	0	0	1	1	1	1	1	0
0	1	0	1	1	0	1	0	0	1
1	0	0	1	0	1	1	0	0	1
1	1	1	1	0	0	0	0	1	0

Условное графическое обозначение элемента выполняют в виде прямоугольника, в левом верхнем углу которого указывают обозначение выполняемой функции. Входы элемента обозначают линиями слева, выходы — справа.



а *б* *в*

Рис. 3.2. Логические элементы:

а — элемент И на два входа; *б* — элемент ИЛИ на два входа; *в* — элемент НЕ

Элементы И, ИЛИ и НЕ составляют функционально полную систему, т.е. из них можно составить любую комбинационную (логическую) схему. Логические элементы, выполненные в виде интегральных схем, обычно реализуют логические функции И—НЕ (штрих Шеффера) или ИЛИ—НЕ (стрелка Пирса). Каждый из этих элементов обладает свойством функциональной полноты. Обозначения элементов И—НЕ и ИЛИ—НЕ на функциональных схемах, а также примеры их реализации приведены на рис. 3.3.

Элементы И на два входа часто используют в качестве ключей (вентилей), которые управляют передачей данных между двумя схемами (рис. 3.4).

При передаче одноразрядных данных (см. рис. 3.4, *а*) один вход элемента И является информационным, а второй — управляющим. На информационный вход поступают одноразрядные дан-

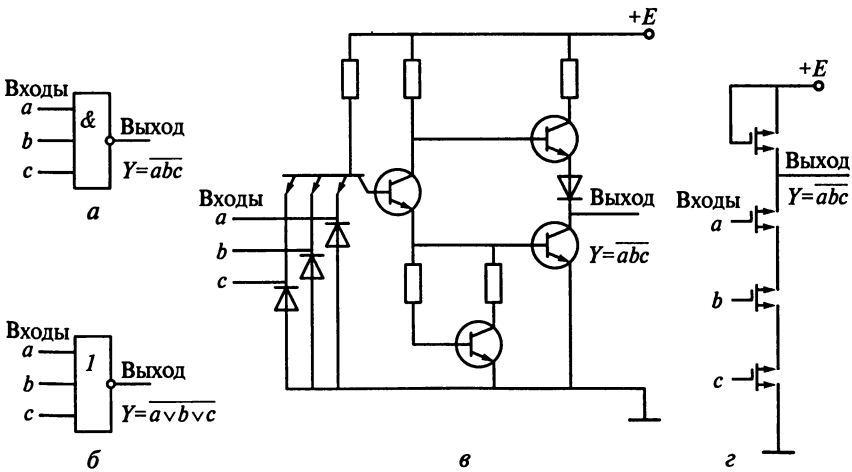


Рис. 3.3. Логические элементы И—НЕ, ИЛИ—НЕ:

а — обозначение элемента И—НЕ на три входа; *б* — обозначение элемента ИЛИ—НЕ на три входа; *в* — принципиальная электрическая схема элемента И—НЕ транзисторно-транзисторной логики; *г* — принципиальная электрическая схема элемента И—НЕ на МОП-транзисторах

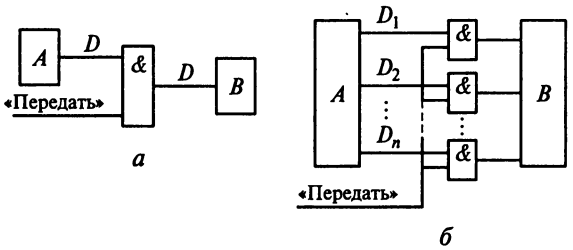


Рис. 3.4. Схемы передачи данных:
а — одноразрядных; *б* — многоразрядных

ные *D* от источника *A*. Если на управляющем входе сигнал «Передать» равен «0», то на приемник *B* поступает сигнал «0» независимо от значения данных *D*. Если сигнал «Передать» равен «1», то сигнал на выходе элемента И совпадает с информационным сигналом и на вход приемника поступают данные *D*.

При передаче многоразрядных данных (см. рис. 3.4, *б*) каждый разряд данных проходит через отдельный элемент И, а сигнал «Передать» подается одновременно на управляющие входы всех ключей.

3.3. Триггеры

Общие сведения о триггерах. Для хранения информации в компьютере могут использоваться различные типы элементов памяти. В зависимости от способа хранения информации элементы памяти могут быть статическими, позволяющими хранить двоичную информацию сколь угодно долго, и динамическими, хранящими информацию в течение ограниченного отрезка времени. Для длительного хранения информации в динамических элементах памяти необходимо периодически восстанавливать ее (регенерировать). Динамические элементы памяти применяют при построении устройств памяти. Здесь рассматриваются статические элементы памяти, которые используют в схемах процессора. В качестве статических элементов памяти в настоящее время применяют триггеры.

Основу триггера составляет бистабильная ячейка, имеющая два устойчивых состояния. Бистабильные ячейки могут быть построены на двух логических элементах И—НЕ или ИЛИ—НЕ, соединенных перекрестными связями (рис. 3.5).

Существование двух устойчивых состояний бистабильной ячейки объясняется наличием в ее схеме обратных связей, позволяющих сигналу с выхода элемента поступать на его же вход через второй элемент. Так, если на рис. 3.5, *а* сигнал на верхнем выходе равен

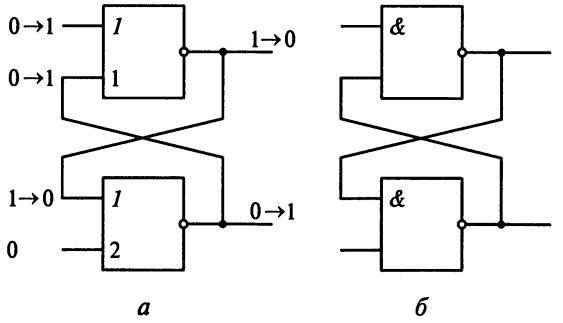


Рис. 3.5. Бистабильная ячейка:
а — на элементах ИЛИ—НЕ; *б* — на элементах И—НЕ

«1» и на оба входа подается сигнал «0», то сигнал «1» с выхода элемента 1 поступает на вход элемента 2 и формирует на его выходе сигнал «0». Этот сигнал поступает на вход элемента 1 и поддерживает такое состояние схемы, делая его устойчивым. В этом состоянии на выходе элемента 1 сигнал равен «1», а на выходе элемента 2 сигнал равен «0».

Для изменения состояния схемы необходимо подать на верхний вход элемента 1 сигнал «1». При этом на выходе элемента 1 сигнал становится равным «0». Тогда на выходе элемента 2 формируется сигнал «1», который вместе с единичным входным сигналом устанавливает выходной сигнал элемента 1 равным «0». Такое состояние схемы также является устойчивым и после того, как сигнал на входе элемента 1 станет равным «0». В этом состоянии на выходе элемента 1 сигнал равен «0», а на выходе элемента 2 — «1». При поступлении сигнала «1» на вход элемента 2 происходит возвращение схемы в начальное состояние. Таким образом, схема имеет два устойчивых состояния, которые можно устанавливать подачей сигнала «1» на вход элемента 1 или 2. Аналогично работает бистабильная ячейка на элементах И—НЕ (см. рис. 3.5, *б*).

Триггер — это цифровая электронная схема с двумя устойчивыми состояниями, которые устанавливаются при подаче соответствующей комбинации входных сигналов и сохраняются.

Кроме бистабильной ячейки в состав триггера входит схема управления (рис. 3.6). *Схема управления* — это комбинационная схема, при помощи которой осуществляется запись информации в триггер (изменение состояний триггера). Конкретный вид схемы управления зависит от типа триггера.

Триггер имеет два выхода: прямой и инверсный (Q и $\bar{Q}$). Сигналы на выходах триггера всегда имеют различные значения. Если на прямом выходе сигнал равен «1», то на инверсном — «0» и наоборот. Состояние триггера определяется значением сигнала на

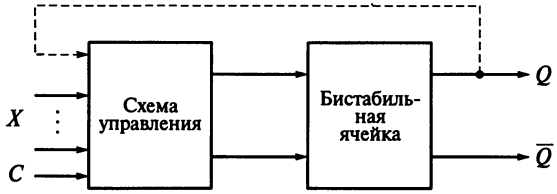


Рис. 3.6. Общая структура триггера

прямом выходе (Q). Если сигнал на прямом выходе равен «1», то триггер находится в состоянии «1». Можно также сказать, что *состояние триггера* — это информация, записанная в триггере. Таким образом, если триггер находится в состоянии «1», то в нем записана единица.

Триггеры могут быть асинхронными или синхронными. В *асинхронных* триггерах используются только основные или информационные входы. Изменение состояния асинхронного триггера может происходить в произвольные моменты времени, определяемые изменениями сигналов на информационных входах.

В *синхронных* триггерах кроме информационных входов имеется вход синхронизации. На этот вход подается сигнал синхронизации C, который выполняет функции сигнала, разрешающего переключение триггера из одного состояния в другое. Если сигнал синхронизации C равен «0», то состояние синхронного триггера не изменяется при любой комбинации сигналов на информационных входах. Для переключения синхронного триггера необходимо подать на информационные входы определенную, зависящую от типа триггера, комбинацию сигналов и, кроме того, установить значение сигнала C, равное «1».

Логика переключения триггера определяется его типом и зависит от числа и назначения входов. Наиболее часто в цифровой технике используют *RS*-, *JK*-, *D*- и *T*-триггеры, а также комбинированные триггеры. Буквами R, S, J, K, D и T обозначаются информационные входы триггеров (X).

Таблица 3.2

Переходы *RS*-триггера

Входы		Состояния	
R	S	0	1
0	0	0	1
0	1	1	1
1	0	0	0
1	1	—	—

Асинхронный *RS*-триггер. Он имеет два информационных входа — R и S. Вход S (*set*) используется для установки триггера в состояние «1», а вход R (*reset*) — в состояние «0», поэтому *RS*-триггер называют *триггером с установочными входами.*

Работу триггера описывает таблица переходов (табл. 3.2).

Входами служат значения входных сигналов R и S, а так-

же значения состояний триггера в текущий момент времени (Q_t). В таблице переходов приведены значения состояний триггера в следующий момент времени (Q_{t+1}). Переходы триггера из одного состояния в другое происходят, если на вход R или S подается сигнал «1».

При $R = 0$ и $S = 0$ состояние триггера не меняется. Такой режим называется *режимом хранения*. В случае если $R = 0$ и $S = 1$ триггер переходит в состояние «1» независимо от того, в каком состоянии он находился до изменения входных сигналов. При $R = 1$ и $S = 0$ триггер переходит в состояние «0». Таким образом, для записи «1» в RS-триггер необходимо подать на его входы сигналы $R = 0$ и $S = 1$, для записи «0» — сигналы $R = 1$ и $S = 0$. Комбинация сигналов $R = 1$ и $S = 1$ является запрещенной, состояние триггера при этом не определено. Таким образом, логика переходов RS-триггера аналогична логике работы электрического клавишного выключателя с двумя положениями: «Вкл» и «Выкл».

Таблица 3.3

Таблица переходов *RS*-триггера

Входы		Текущее состояние	Следующее состояние
R	S	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	Запрещенная комбинация
1	1	1	

Таблица переходов триггера может быть интерпретирована как таблица истинности комбинационной схемы, в которой значения сигналов на входах R_t, S_t и значение текущего состояния Q_t можно рассматривать как логические переменные, а Q_{t+1} — как логическую функцию (табл. 3.3).

RS-триггер может быть построен на различных логических элементах. Функциональная схема асинхронного RS-триггера, построенного на элементах И—НЕ и ИЛИ—НЕ, а также его УГО показаны на рис. 3.7.

Условное графическое обозначение триггера кроме основного поля включает в себя дополнительное. В основном поле указывается назначение элемента (триггер), а в дополнительном — обозначение входов, т.е. тип триггера. Инверсный выход триггера отмечается кружком. Инверсными могут быть и входные сигналы. Так, при построении RS-триггера на элементах И—НЕ действующими (активными) являются инверсные значения входных сигналов R и S (сигналы «0»). Это означает, что для переключения триггера из одного состояния в другое нужно подать сигнал «0» на соответствующий вход триггера (R или S). Инверсные входы на схемах также отмечаются кружком. Заметим, что таблицы переходов триг-

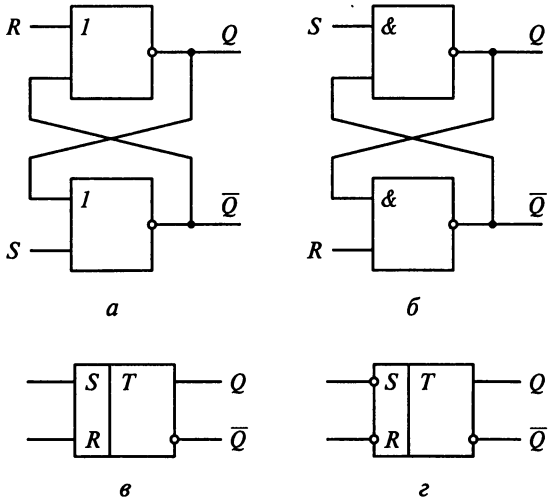


Рис. 3.7. Асинхронный RS-триггер:

а — на элементах ИЛИ—НЕ; *б* — на элементах И—НЕ; *в* — УГО асинхронного *RS*-триггера с прямыми входами; *г* — УГО асинхронного *RS*-триггера с инверсными входами

геров обычно приводятся для прямых значений входных сигналов. Асинхронный *RS*-триггер представляет собой бистабильную ячейку, поэтому он используется как основа при построении всех триггеров.

Синхронный *RS*-триггер. Этот триггер имеет дополнительно вход C, на который поступает синхросигнал. Информационные сигналы R и S могут изменять состояние триггера только при значении синхросигнала $C = 1$. Таблица переходов синхронного *RS*-триггера состоит из двух частей. Первая часть таблицы описывает переходы триггера при $C = 1$ и совпадает с таблицей переходов асинхронного триггера (см. табл. 3.2). Когда $C = 0$, триггер не меняет своего состояния при любой комбинации сигналов на информационных входах и логика его переходов может быть описана табл. 3.4.

Таблица 3.4

Переходы синхронного *RS*-триггера

Входы			Состояния	
C	R	S	0	1
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1

Отметим, что при $C = 0$ разрешенными являются любые комбинации входных сигналов, в том числе $R = 1$, $S = 1$.

На рис. 3.8 приведены функциональные схемы синхронных *RS*-триггеров, реализованных на элементах И—НЕ и

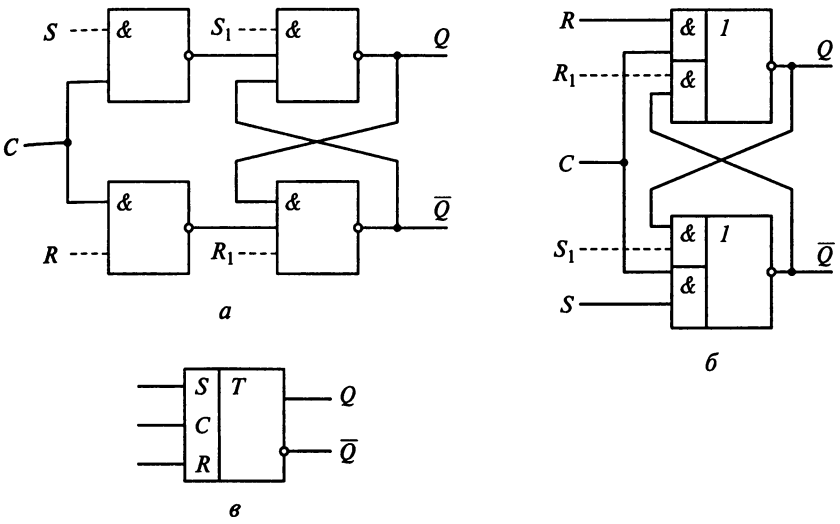


Рис. 3.8. Синхронный *RS*-триггер:
а — на элементах И—НЕ; *б* — на элементах И—ИЛИ—НЕ; *в* — УГО

И—ИЛИ—НЕ, и их условное графическое обозначение. Кроме основных входов R и S там показаны дополнительные входы R_1 и S_1, которые являются асинхронными. При подаче сигналов на них состояние триггера может изменяться независимо от значения сигнала C. Следует отметить, что в каждый момент времени можно управлять переходами триггера только с помощью синхронных или асинхронных входов.

Двухтактный *RS*-триггер. Триггеры используются в различных узлах ЭВМ, между которыми осуществляется передача информации. Устойчивая работа цепочки триггеров происходит только в том случае, когда запись новой информации в триггер производится после считывания прежней и передачи ее в следующий по цепочке триггер.

Это возможно при использовании двух серий синхроимпульсов, сдвинутых относительно друг друга на полпериода. Такой принцип управления и синхронизации применяется в двухтактных триггерах. Двухтактные триггеры используются в сдвигающих регистрах, а также в качестве элементов памяти в цифровых автоматах с памятью для устранения эффекта гонок.

Простейшая схема двухтактного *RS*-триггера может быть построена из двух однотактных, причем синхросигналы на входы C первого и второго триггеров должны подаваться в противофазе. Это делается с помощью инвертора (рис. 3.9, *а*).

В основном поле условного графического обозначения двухтактного триггера записываются две буквы T (рис. 3.9, *б*). Особен-

ности переключения двухтактного триггера из одного состояния в другое поясняются временно́й диаграммой (рис. 3.9, *в*).

Пусть оба триггера находятся в состоянии «0» и на входы триггера поступают сигналы $S = 1$ и $R = 0$ (запись в триггер сигнала «1»). При поступлении на вход *RS*-триггера сигнала $C = 1$ входная информация по переднему фронту сигнала *C* запоминается в первом однотактном триггере (он переходит в состояние «1»). Второй однотактный триггер хранит информацию о предыдущем состоянии, так как на его входе $C = 0$.

По окончании действия синхросигнала (по заднему фронту), т.е. при $C = 0$, первый триггер переходит в режим хранения, а информация с выходов первого триггера передается на вход второго триггера. Так как на входе второго триггера сигнал $C = 1$, он также переходит в состояние «1». В результате к началу следующего такта на выходе двухтактного *RS*-триггера появится сигнал состояния, соответствующего входной информации. Аналогичным образом производится запись в двухтактный триггер сигнала нуля.

Для установки *RS*-триггера в «0» или «1» независимо от присутствия сигнала на входе *C* в схему вводят прямые или инверсные

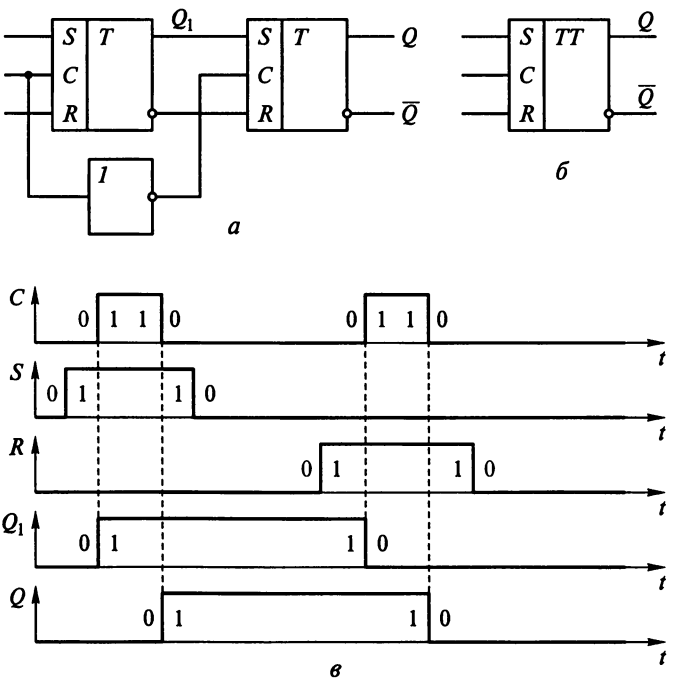


Рис. 3.9. Двухтактный *RS*-триггер:
а — схема; *б* — УГО; *в* — временна́я диаграмма работы

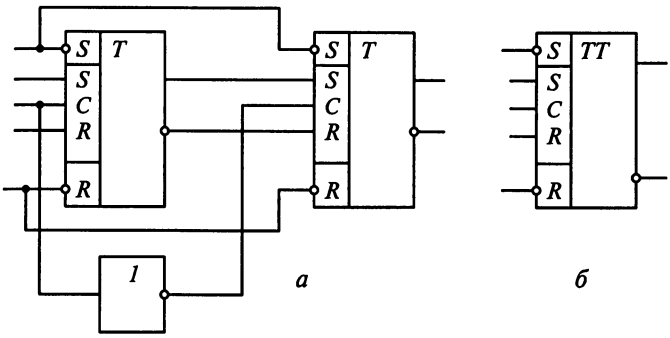


Рис. 3.10. Двухтактный *RS*-триггер с дополнительными входами *R* и *S*: *а* — схема; *б* — УГО

входы *R* и *S* асинхронной установки (рис. 3.10, *а*) и отображают их на УГО (рис. 3.10, *б*).

Асинхронный и синхронный *D*-триггеры. В вычислительной технике широко применяют *D*-триггеры, которые реализуют функцию временно́й задержки входного сигнала. Также *D*-триггеры имеют один информационный вход. Логика работы асинхронного *D*-триггера описывается таблицей переходов (табл. 3.5).

В асинхронном D-триггере состояние (выходной сигнал) Q_{t+1} повторяет значение входного сигнала D_t, поэтому асинхронный *D*-триггер по существу не является элементом памяти и рассматривается только как основа для построения синхронного *D*-триггера.

Функциональная схема и УГО синхронного *D*-триггера, построенного на основе синхронного *RS*-триггера, показаны на рис. 3.11. Для преобразования *RS*-триггера в *D*-триггер сигнал *D* подается на вход *S* непосредственно, а на вход *R* — через инвертор. Если при $C = 1$ на вход *D* подать сигнал «1», то триггер перейдет в состояние «1», а при подаче сигнала $D = 0$ в триггер будет записан

Таблица 3.5

Переходы асинхронного *D*-триггера

Вход	Состояния	
D	0	1
0	0	0
1	1	1

Таблица 3.6

Переходы синхронного *D*-триггера

Входы		Состояния	
D	*C*	0	1
0	0	0	1
0	1	0	0
1	0	0	1
1	1	1	1

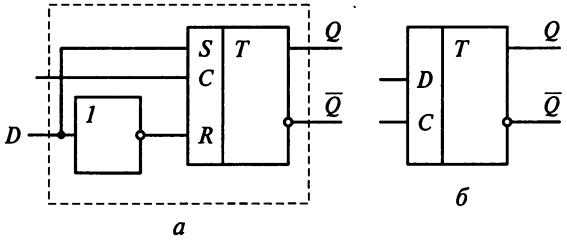


Рис. 3.11. Синхронный *D*-триггер:
а — схема; *б* — УГО

«0». Таким образом, для записи в *D*-триггер единицы на вход *D* нужно подать сигнал «1», а для записи нуля — сигнал «0» (так как триггер синхронный, на вход *C* необходимо в обоих случаях подавать сигнал «1»). Это делает *D*-триггер удобным для использования в схемах статической памяти, так как для записи достаточно иметь одну линию на разряд данных. При этом сигнал *C* является общим для всех разрядов записываемых данных.

Логику работы синхронного *D*-триггера описывает табл. 3.6. Эту логику можно охарактеризовать выражением «что надо записать в *D*-триггер, то и подается на его вход».

Наличие входа синхронизации позволяет записывать новые данные в триггер только в определенные моменты времени (при $C = 1$). В промежутках между ними данные в триггере сохраняются без изменения. При чтении данных из триггера его состояние также не меняется.

***T*-триггер.** Этот триггер имеет один информационный вход. Логику работы асинхронного *T*-триггера характеризует таблица переходов (табл. 3.7).

При $T = 1$ асинхронный *T*-триггер меняет свое состояние на противоположное, а при $T = 0$ состояние триггера не изменяется. (Аналогичную логику работы имеет кнопочный выключатель на-

Таблица 3.7

Переходы асинхронного *T*-триггера

Входы	Состояния	
T	0	1
0	0	1
1	1	0

Таблица 3.8

Переходы синхронного *T*-триггера

Входы		Состояния	
C	*T*	0	1
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

стольной лампы, который меняет состояние лампы при каждом нажатии кнопки.)

Так как T-триггер суммирует (или подсчитывает) по модулю два числа единиц, поступающих на его информационный вход, то T-триггер называют также *триггером со счетным входом.*

Логику работы синхронного T-триггера описывает табл. 3.8.

При $C = 0$ триггер не изменяет своего состояния, а при $C = 1$ работает как асинхронный T-триггер.

Функциональная схема T-триггера может быть построена на основе синхронного RS-триггера (однотактного или двухтактного). Схемы асинхронного и синхронного T-триггеров показаны на рис. 3.12 и 3.13 соответственно.

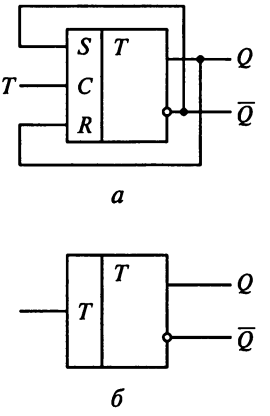


Рис. 3.12. Асинхронный T-триггер:
а — схема; *б* — УГО

Поскольку на этих схемах сигнал с выхода триггера поступает на его же вход, триггер должен во время переключения сохранять состояние и одновременно воспринять новую информацию. Для устойчивой работы в этом случае целесообразно использовать двухтактные триггеры.

***JK*-триггер.** Такие триггеры называют *универсальными.* Универсальность схемы JK-триггера состоит в том, что простой коммутацией входов и выходов можно получать схемы других типов триггеров.

JK-триггер имеет два информационных входа. Вход J используется для установки триггера в состояние «1», а вход K — в состояние «0», т.е. входы J и K аналогичны входам S и R RS-триггера. Отличие JK-триггер от RS-триггера заключается в том, что на входы J и K могут одновременно поступать сигналы «1». В этом случае JK-триггер изменяет свое состояние. Таким образом, он работает так же, как RS-триггер, за исключением комбинации сигналов $J = 1$; $K = 1$, при которой он работает как T-триггер.

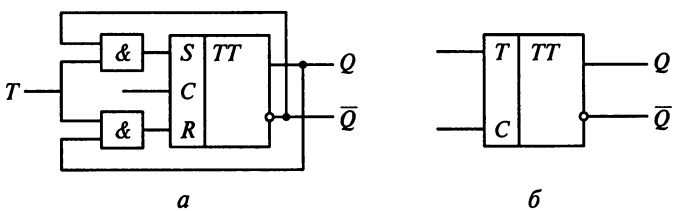
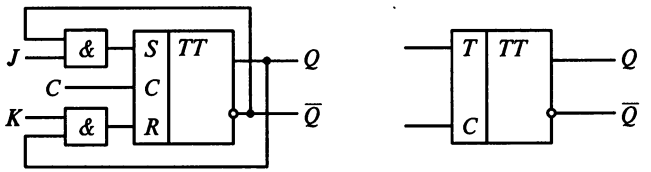
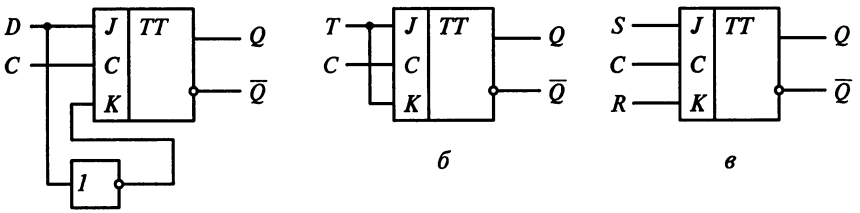


Рис. 3.13. Синхронный T-триггер:
а — схема; *б* — УГО



а *б*

Рис. 3.14. Двухтактный *JK*-триггер:
а — схема; *б* — УГО



а *б* *в*

Рис. 3.15. Схемы преобразования *JK*-триггера:
а — в *D*-триггер; *б* — в *T*-триггер; *в* — *RS*-триггер

При $C = 1$ переходы *JK*-триггера описывает табл. 3.9.

Функциональная схема двухтактного *JK*-триггера и его УГО показаны на рис. 3.14. Этот триггер представляет собой комбинацию *RS*- и *T*-триггеров, что согласуется с логикой его работы. Примеры построения других типов триггеров на основе *JK*-триггера представлены на рис. 3.15. Следует отметить, что триггер любого типа можно преобразовать в любой другой триггер.

Таблица 3.9

Переходы *JK*-триггера

Входы		Состояния	
J	*K*	0	1
0	0	0	1
0	1	0	0
1	0	1	1
1	1	1	0

Таблица 3.10

Логика работы дешифратора

Входы		Выходы			
a	*b*	f_0	f_1	f_2	f_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

3.4. Типовые узлы комбинационного типа

Дешифраторы. Дешифраторы *DC* имеют несколько входов (n) и несколько выходов (N) и предназначены для преобразования входного кода в сигнал только на одном из выходов. Обычно $N = 2^n$. Такие дешифраторы называются *полными*. Входной сигнал рассматривается как двоичное число. При поступлении числа на входы дешифратора только на одном его выходе, номер которого равен числу на входе, выдается сигнал «1», а на остальных выходах — сигнал «0». Нумерация выходов начинается с «0». Например, если дешифратор имеет три входа и на него поступает сигнал «101», то на пятом выходе возникнет сигнал «1», а на остальных выходах — «0». Дешифраторы используются, например, в устройствах памяти для выбора заданной ячейки по ее адресу.

Логику работы дешифратора на два входа описывает табл. 3.10.

В соответствии с таблицей истинности логические функции выходов дешифратора имеют следующий вид: $f_0 = \overline{a}\overline{b}$; $f_1 = \overline{a}b$; $f_2 = a\overline{b}$; $f_3 = ab$. Функциональная схема дешифратора на два входа с инверсными выходами, выполненная на элементах И—НЕ, и его УГО приведены на рис. 3.16. Наличие инверсных выходов означает, что на одном из выходов дешифратора сигнал равен нулю, а на всех остальных — единице.

Такой дешифратор состоит из нескольких одинаковых схем, не связанных между собой, и называется *линейным*. При большом числе входов дешифраторы имеют более сложную структуру (пирамидальные и ступенчатые дешифраторы).

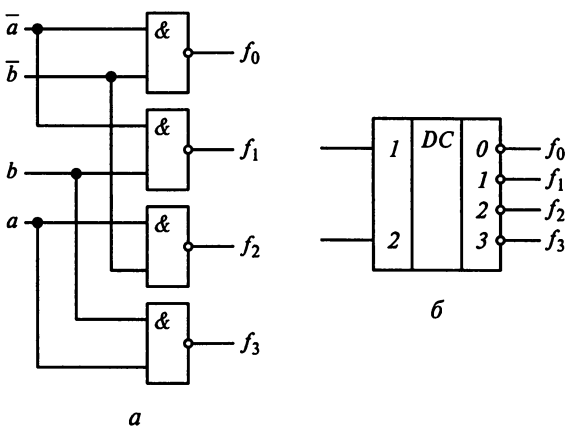


Рис. 3.16. Дешифратор на два входа:
а — функциональная схема; *б* — УГО

Из дешифраторов с некоторым числом входов можно построить дешифратор на большее число входов. Каскадный принцип построения таких дешифраторов поясняется на рис. 3.17, где дешифратор с четырьмя входами выполнен на синхронизируемых двухвходовых дешифраторах с инверсными выходами. Схема такого дешифратора состоит из двух ступеней. Первую ступень составляет дешифратор на два входа, на который поступают два из четырех входных сигналов (c, d). Выходные сигналы первой ступени разрешают работу одного из четырех дешифраторов второй ступени, на основные входы которых поступают остальные входные сигналы (a, b). В каждый момент времени в зависимости от значений сигналов (c, d) работает один из дешифраторов второй ступени. Совместная работа всех четырех дешифраторов позволяет формировать один из 16 выходных сигналов в соответствии с входным сигналом (a, b, c, d).

Входы синхронизации C являются инверсными, т. е. при $C = 1$ на всех выходах дешифратора сигналы равны единице, а при $C = 0$ только на одном из выходов сигнал равен нулю, так как дешифратор имеет инверсные выходы.

Одноразрядный сумматор. Сумматоры выполняют арифметическое сложение чисел, которое производится начиная с младших

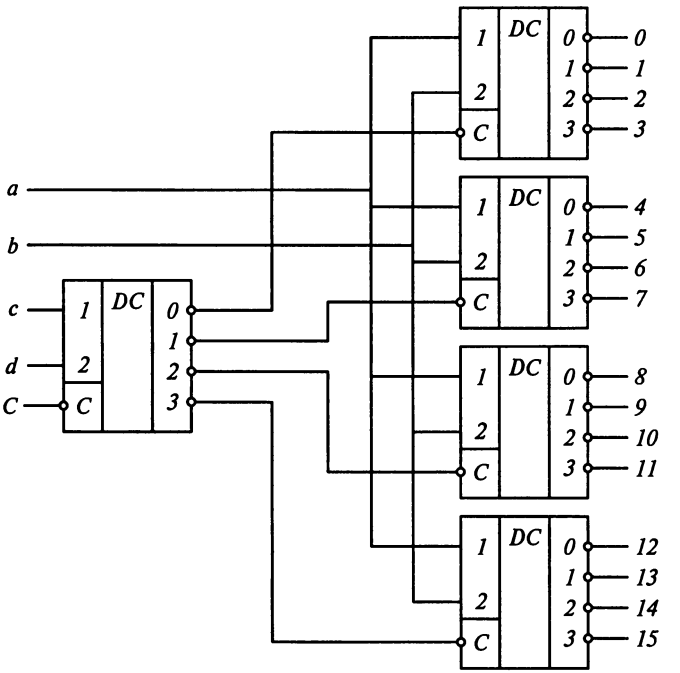


Рис. 3.17. Каскадный дешифратор

разрядов чисел. В каждом разряде сумматора выполняются одинаковые действия, т. е. суммируются две двоичные цифры в соответствии с правилами сложения двоичных цифр:

0 0 1 1 — разряды первого слагаемого;

+ + + +

0 1 0 1 — разряды второго слагаемого;

– – – –

0 1 1 10 — разряды суммы (переноса).

При сложении двух единиц возникает единица переноса в старший разряд, которую нужно учесть при сложении цифр следующего разряда. Поэтому в каждом разряде сумматора необходимо предусмотреть возможность суммирования трех цифр: двух цифр слагаемых и единицы переноса из младшего разряда. В свою очередь, в каждом разряде необходимо сформировать не только значение одного разряда суммы, но и значение единицы переноса в соседний старший разряд. Таким образом, сумматор для сложения многоразрядных чисел можно построить из одинаковых схем, каждая из которых выполняет сложение двух цифр слагаемых и переноса из младшего разряда. Такая схема называется *одноразрядным сумматором*. Одноразрядный сумматор представляет собой комбинационную схему с тремя входами и двумя выходами, логика работы которой соответствует следующей таблице истинности (табл. 3.11).

Таблица 3.11

Логика одноразрядного сумматора

Входы			Выходы	
a_i	b_i	p_{i-1}	s_i	p_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Примечание. В таблице приняты следующие обозначения: a_i — цифра i-го разряда первого слагаемого; b_i — цифра i-го разряда второго слагаемого; p_{i-1} — перенос из разряда $i-1$; s_i — значение суммы в разряде i; p_i — значение переноса из разряда i.

В соответствии с таблицей истинности уравнения выходов после минимизации имеют следующий вид:

$$s_i = a_i b_i p_{i-1} \vee \bar{a}_i \bar{b}_i p_{i-1} \vee \bar{a}_i b_i \bar{p}_{i-1} \vee a_i \bar{b}_i \bar{p}_{i-1};$$
$$p_i = a_i b_i \vee b_i p_{i-1} \vee a_i p_{i-1}.$$

Схемы одноразрядного сумматора и его УГО показаны на рис. 3.18.

Полусумматор. *Полусумматором*, или *одноразрядным сумматором на два входа*, называется схема с двумя входами и двумя выходами, которая реализует следующие функции:

$$M_i = a_i \oplus b_i;\ F_i = a_i b_i,$$

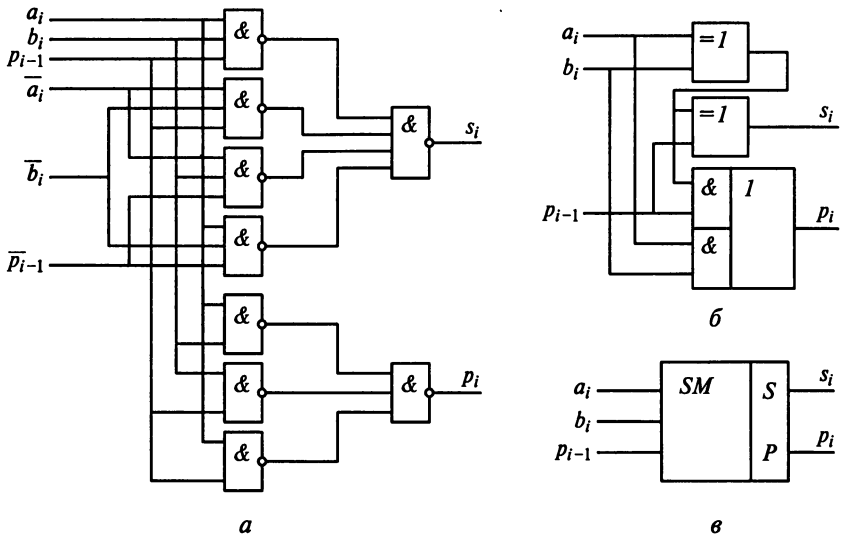


Рис. 3.18. Одноразрядные сумматоры:

а — схема сумматора на элементах И—НЕ; *б* — схема сумматора с использованием элементов сложения по модулю 2; *в* — УГО

где M_i — значение суммы в данном разряде; F_i — значение переноса из данного разряда; a_i и b_i — одноименные разряды слагаемых.

Полусумматоры могут быть использованы для построения схем одноразрядных сумматоров на три входа, а также схем ускорения умножения. Варианты схемы полусумматора и его УГО приведены на рис. 3.19.

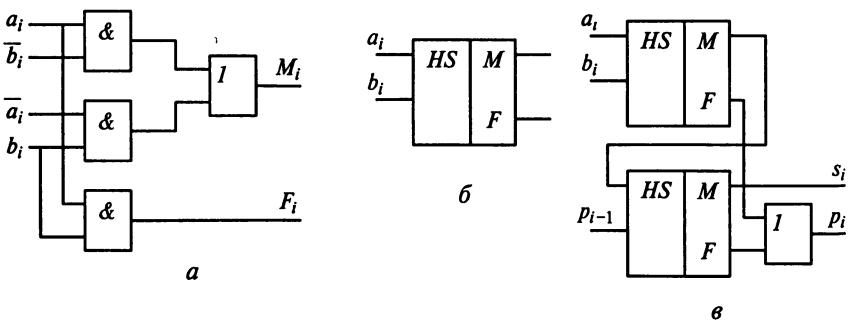


Рис. 3.19. Полусумматор:

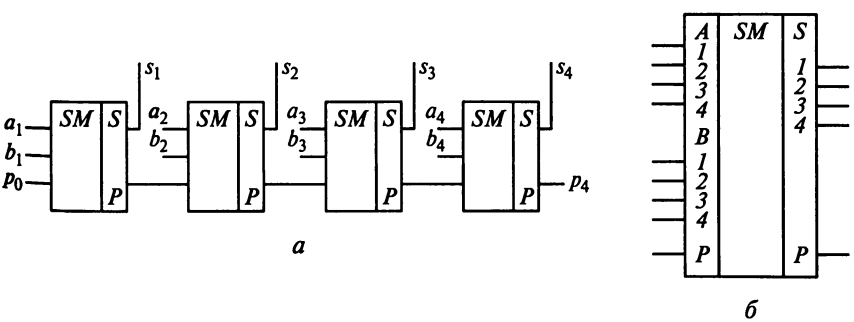
а — схема; *б* — УГО; *в* — схема одноразрядного сумматора на основе полусумматоров

Многоразрядные сумматоры. Они строятся из одноразрядных сумматоров. Число одноразрядных сумматоров равно разрядности слагаемых. При этом одноразрядные сумматоры связаны между собой только цепями переносов. Схема четырехразрядного сумматора показана на рис. 3.20.

На входы a_i и b_i подаются соответствующие разряды слагаемых, на вход p_{i-1} поступает перенос из соседнего младшего разряда сумматора. Значение суммы в данном разряде подается на выход s_i, значение переноса p_i — в соседний старший разряд.

Недостатком схемы рис. 3.20 является большое время суммирования, так как единица переноса может проходить последовательно через все разряды сумматора (например, при сложении чисел типа 01111111 и 00000001). Такая схема называется *сумматором с последовательными переносами*. Для уменьшения времени задержки используют сумматоры с параллельными (рис. 3.21), сквозными или групповыми переносами.

Сумматор с параллельными переносами состоит из схемы формирования суммы и схемы ускоренных переносов. *Схема формирования суммы* (верхняя часть рис. 3.21) выдает сигналы разрядных сумм. *Схема ускоренных переносов* вырабатывает сигналы переноса одновременно во всех разрядах. Идея построения таких схем заключается в том, что сигнал переноса должен обходить группы разрядов сумматора, в которых значение суммы или переноса равно единице. Это достигается за счет анализа значений слагаемых как в текущем разряде сумматора, так и во всех младших разрядах. Для упрощения схемы вводят вспомогательные функции формирования и распространения переносов. Функция формирования переноса F_i описывает условие возникновения переноса в разряде i: $F_i = a_i b_i$. Функция распространения переносов R_i определяет условие, при котором сигнал переноса из младшего разряда передается в соседний старший разряд: $R_i = a_i \vee b_i$. Тогда перенос из данно-



а *б*

Рис. 3.20. Четырехразрядный двоичный сумматор:
а — схема; *б* — УГО

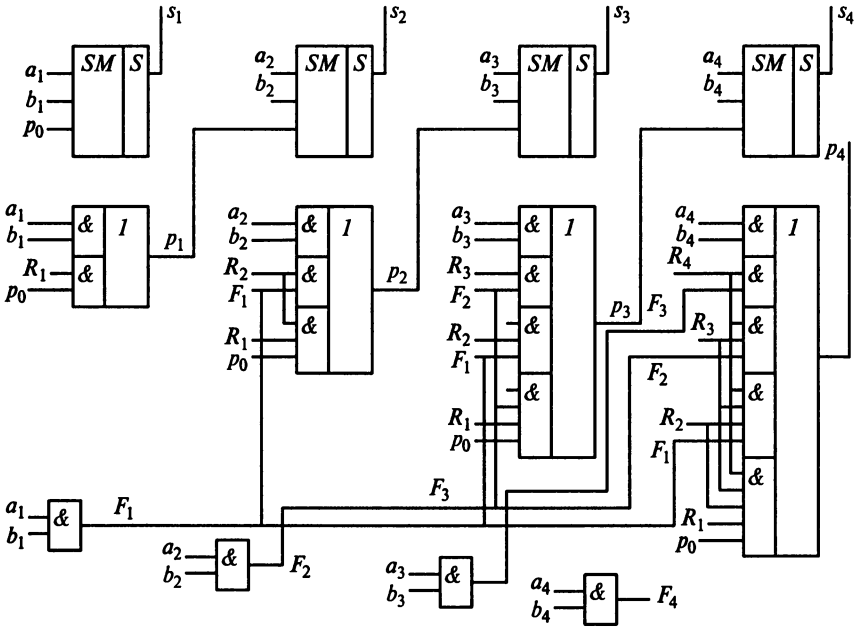


Рис. 3.21. Сумматор с параллельными переносами

го разряда передается, если $p_i = F_i \vee R_i p_{i-1} = 1$. Схема ускоренных переносов (см. рис. 3.21) построена с учетом рекурсивного характера функций p_i. Сигнал переноса формируется одновременно, при этом он во всех разрядах проходит через три логических элемента, и, следовательно, задержка сигнала переноса не зависит от разрядности сумматора.

Сумматор с параллельными переносами требует значительных аппаратурных затрат, причем сложность схемы сильно возрастает с увеличением разрядности сумматора. Поэтому многоразрядные сумматоры могут выполняться с групповыми переносами. Для этого сумматор разбивают на группы разрядов. Переносы в группах и между группами организуют параллельно, последовательно или по принципу сквозных переносов (рис. 3.22).

В сумматорах со сквозными переносами сигнал переноса формируется в соответствии с функцией $p_i = R_i p_{i-1} = (a_i \vee b_i) p_{i-1}$. При этом сигнал переноса проходит в каждом разряде только через один элемент И, что меньше, чем в сумматоре с последовательными переносами.

Таким образом, сумматор со сквозными переносами занимает промежуточное положение между сумматорами с параллельными и последовательными переносами по быстродействию и аппаратным затратам.

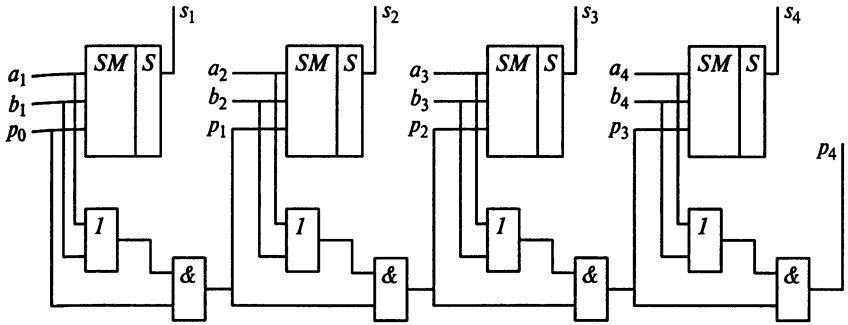


Рис. 3.22. Сумматор со сквозными переносами

Двоично-десятичные сумматоры. При обработке больших массивов десятичных чисел значительная часть времени расходуется на перевод чисел из одной системы счисления в другую. В этом случае целесообразно выполнять обработку данных непосредственно в десятичной системе счисления. При этом для представления десятичных чисел используют различные двоично-десятичные коды. Десятичные цифры представляются двоичными тетрадами. Сложение тетрад выполняется с помощью двоично-десятичных сумматоров.

Двоично-десятичный сумматор строится на основе четырехразрядного двоичного сумматора, в котором перенос возникает, если значение суммы равно или больше 16. Но при сложении двух десятичных цифр перенос должен возникать, если их сумма равна или больше 10, поэтому для правильного сложения двоично-десятичных цифр двоичный сумматор дополняется схемой коррекции. Коррекция выполняется для каждой тетрады суммы отдельно. Правила коррекции зависят от используемого двоично-десятичного кода.

Для кода прямого замещения коррекция суммы выполняется по следующим правилам:

1. Если двоичная сумма не более 9, то коррекция не требуется.
2. Если двоичная сумма принимает значение от 10 до 15, необходимо искусственно вызвать перенос в следующую тетраду. Для этого коррекция выполняется путем прибавления к тетраде десятичного числа 6 или двоичного 0110.
3. Если двоичная сумма принимает значение от 16 до 19, то возникает перенос из тетрады, который имеет вес, равный 16, и уменьшает значение суммы на 6. Коррекция выполняется так же, как и во втором случае, прибавлением двоичного числа 0110 к данной тетраде.

Таким образом, при сложении двоично-десятичных чисел в коде прямого замещения коррекция выполняется в тех тетрадах,

в которых возникла запрещенная комбинация или из которых возник перенос.

При сложении чисел со знаком используются обратный или дополнительный код. Для получения ОК к каждой тетраде прибавляется двоичное число 0110, а затем все цифровые тетрады инвертируются. Дополнительный код получают из обратного путем прибавления единицы к младшей тетраде.

Пример 3.1. Определить сумму чисел в двоично-десятичном коде, если $A_{10} = -183$; $B_{10} = 924$.

$A_{2-10} =$	$-\ 0001\ 1000\ 0011;$	$B_{2-10} = +\ 1001\ 0010\ 0100$
$(A_{2-10})^{ПК} =$	$1\ 0001\ 1000\ 0011$	$(B_{2-10})^{ПК} = 0\ 1001\ 0010\ 0100$
	$+\ 0110\ 0110\ 0110$	Прибавление кода 0110
$(A_{2-10})^{ОК} =$	$1\ 0111\ 1110\ 1001$	
	$1\ 1000\ 0001\ 0110$	Инверсия
	$+\qquad\qquad 0001$	Прибавление единицы
$(A_{2-10})^{ДК} =$	$1\ 1000\ 0001\ 0111$	$(B_{2-10})^{ДК} = 0\ 1001\ 0010\ 0100$
$(B_{2-10})^{ДК} =$	$+\ 0\ 1001\ 0010\ 0100$	Сложение
$(A_{2-10} + B_{2-10})^{ДК} =$	$10\ 0001\ 0011\ 1011$	При суммировании возник
коррекция	$+\quad 0110\qquad 0110$	перенос из первой тетрады
$(A_{2-10} + B_{2-10})^{ДК} =$	$0\ 0111\ 0100\ 0001;$	и образовалась запрещенная комбинация в последней
$(A + B)_{2-10} =$	$+\quad 0111\ 0100\ 0001;$	тетраде
$(A + B)_{10} =$	$+\quad 741.$	

Одноразрядный двоично-десятичный сумматор содержит два четырехразрядных двоичных сумматора и схему коррекции (рис. 3.23). Первый двоичный сумматор выполняет сложение двоично-десятичных тетрад как сложение двоичных чисел. Схема коррекции формирует сигнал коррекции при возникновении переноса из тетрады или появлении запрещенной комбинации в тетраде двоичной суммы. Признаком запрещенной комбинации ($S_4S_3S_2S_1$ = 1010, 1011, 1100, 1101, 1110, 1111) является наличие единиц одновременно в разрядах суммы S_4S_2 или S_4S_3. Вместе с двоичной суммой сигнал коррекции поступает на входы b_2 и b_3 второго двоичного сумматора, при этом к двоичной сумме добавляется код коррекции, равный 0110_2.

Мультиплексоры. Мультиплексор представляет собой комбинационную схему с несколькими входами и одним выходом. Входы мультиплексора делятся на информационные и управляющие (адресные). Мультиплексор передает данные с одного из информационных входов на выход. Номер (адрес) подключаемого входа задается на управляющих входах. Мультиплексор с k управляющи-

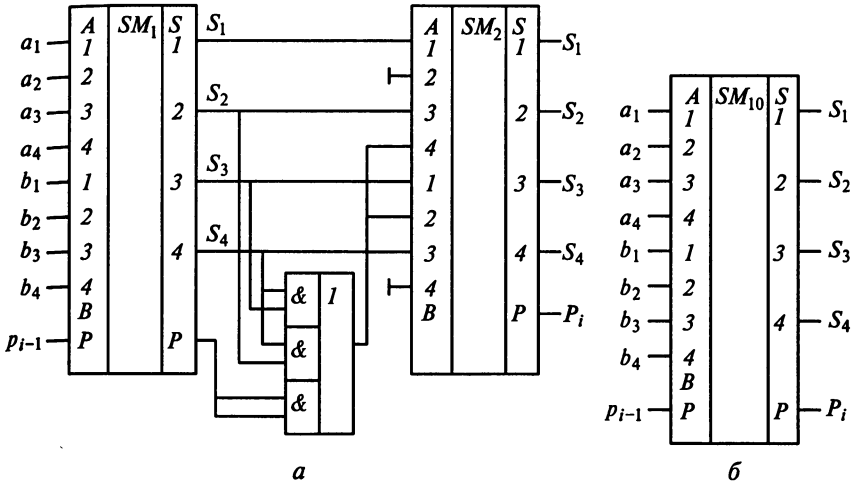


Рис. 3.23. Одноразрядный двоично-десятичный сумматор:
а — схема; *б* — УГО

ми входами может иметь до 2^k информационных входов. Для выбора подключаемого входа используется дешифратор. Функциональная схема мультиплексора с двумя управляющими (A_0 и A_1) и четырьмя информационными ($D_0 \ldots D_3$) входами показаны на рис. 3.24, *а*. Схема мультиплексора включает в себя дешифратор на два входа и выходную схему. При поступлении адреса A_1A_0 на входы

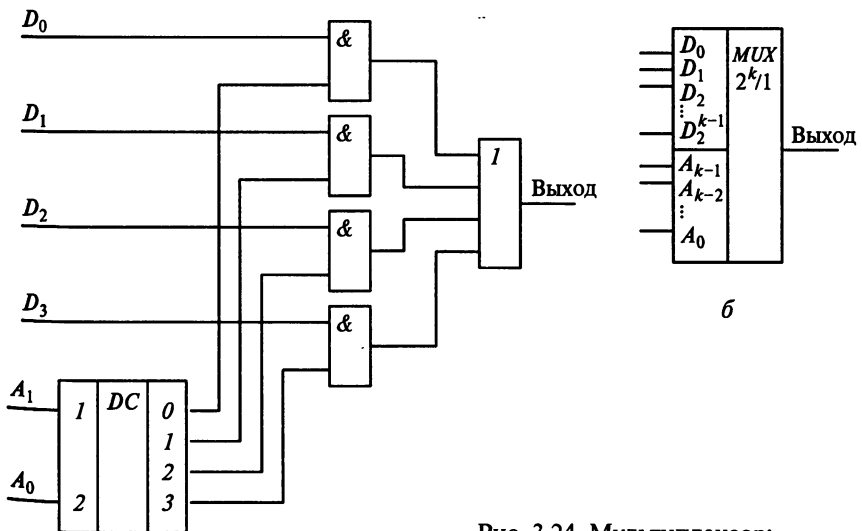


Рис. 3.24. Мультиплексор:
а — функциональная схема; *б* — УГО

дешифратора на одном из его выходов формируется сигнал «1», который подключает вход D_i с заданным адресом к выходу схемы. Условное графическое обозначение мультиплексора приведено на рис. 3.24, *б*.

Используя теорему разложения булевой функции, на мультиплексорах можно реализовать любую логическую функцию. Из мультиплексоров с небольшим числом входов можно построить мультиплексор с необходимым числом входов, используя каскадные схемы (рис. 3.25).

Схема (см. рис. 3.25, *а*) реализует таблицу истинности элемента И—НЕ на два входа, так как на входах схемы зафиксированы значения функции И—НЕ, а на адресные входы подаются значения

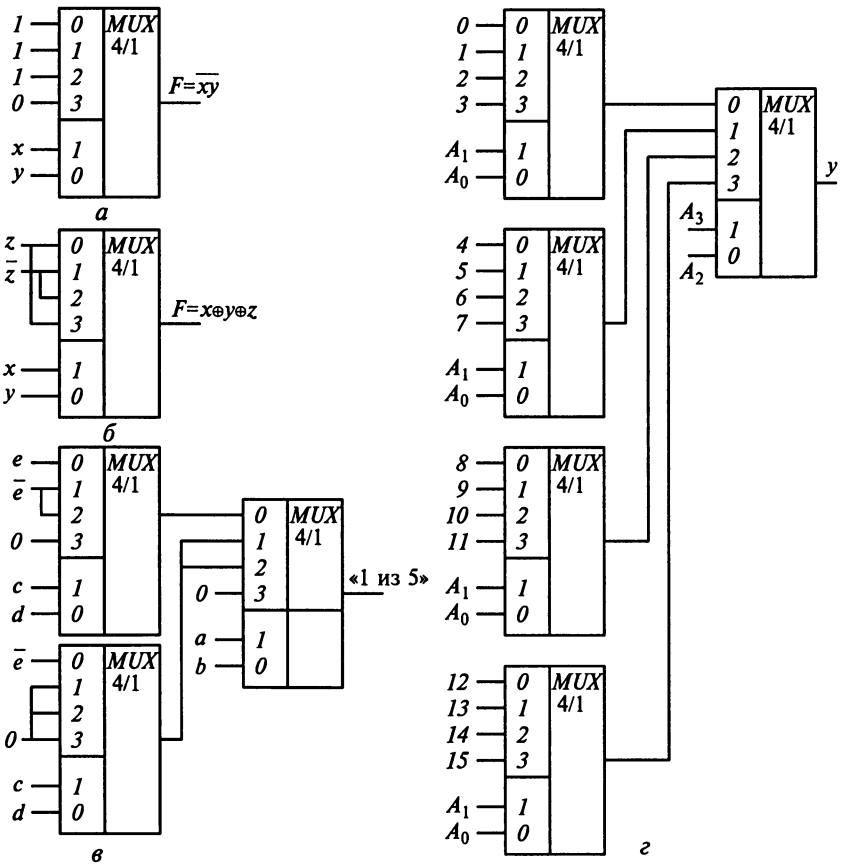


Рис. 3.25. Схемы реализации логических функций:

а — функция И—НЕ; *б* — сложение по модулю 2; *в* — выделение наборов пяти переменных, содержащих одну единицу; *г* — каскадный мультиплексор на 16 входов

переменных x и y. Для выполнения операции сложения переменных x, y и z по модулю 2 входные сигналы подаются так, как это показано на рис. 3.25, *б*. Если, например, $x = 0$, $y = 1$, $z = 0$, то на адресные входы поступит адрес «01», будет выбран вход *1*, и на выход будет выдан сигнал $F = z = 1$, т.е. $F = x \oplus y \oplus z = 0 \oplus 1 \oplus 0 = 1$. Если на вход схемы (см. рис. 3.25, *в*) подать комбинацию сигналов $abcde$ = 00100, то cd = 10 и, следовательно, на мультиплексорах первого яруса будет выбран вход *2*. При этом на вход *0* мультиплексора второго яруса поступит сигнал e, а на входы *1* и *2* — сигнал «0». Так как $ab = 00$, будет выбран нулевой вход этого мультиплексора. На выходе схемы возникнет сигнал $e = 1$, т.е. будет обнаружен код, содержащий одну единицу. Пример каскадного мультиплексора на 16 входов приведен на рис. 3.25, *г*.

Демультиплексоры. Демультиплексор представляет собой комбинационную схему с одним информационным входом, несколькими управляющими (адресными) входами и несколькими выходами. Демультиплексор передает сигнал с информационного входа на один из выходов, номер (адрес) которого задается сигналом на адресных входах. Таким образом, демультиплексор выполняет функцию, обратную по отношению к функции мультиплексора. Так как мультиплексор и демультиплексор выполняют функции, связанные с выбором одного из входов или выходов, их называют также *селекторами*. Максимальное число выходов демультиплексора составляет 2^k, где k — число адресных входов. Функциональная схема демультиплексора с четырьмя информационными выходами и его УГО показаны на рис. 3.26. Входной сигнал D подает-

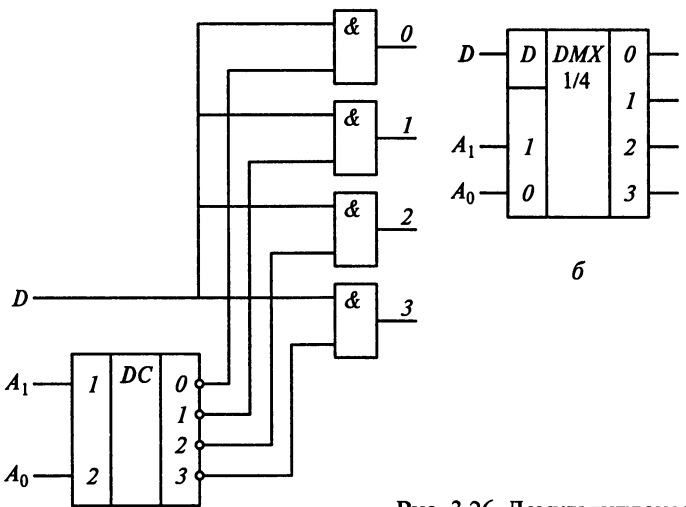


Рис. 3.26. Демультиплексор:
а — функциональная схема; *б* — УГО

ся на входы всех выходных ключей, выполненных на элементах И. На вторые входы ключей поступают сигналы с выхода дешифратора, которые открывают один из ключей и разрешают входному сигналу пройти на выход схемы с заданным адресом.

3.5. Типовые узлы накапливающего типа

Регистры. Регистры предназначены для приема, временного хранения и выдачи данных. Чаще всего данные сохраняются в регистре на время выполнения одной или нескольких машинных команд. Кроме хранения данных регистры могут выполнять и другие операции (сдвиг данных, логические операции). Основу регистра составляют триггеры, число которых равно разрядности хранимых данных.

Наиболее простую схему имеет регистр с параллельной записью данных. На рис. 3.27 приведена схема регистра, построенного на синхронных *D*-триггерах с дополнительными асинхронными входами *S* и *R*. Записываемые данные ($D_0 \ldots D_{n-1}$) подаются на информационные входы триггеров. Запись производится при поступлении сигнала «Прием», подаваемого в определенный момент времени на входы синхронизации всех триггеров регистра. Для установки регистра в состояние «0» используется сигнал «Уст. 0», который подается на дополнительные входы триггеров *R*.

Данные из регистра с параллельной записью и входом установки в состояние «0» могут быть выданы в прямом или обратном (инверсном) коде (рис. 3.28, *а*). Условное графическое обозначение регистра показано на рис. 3.28, *б*.

Для выдачи данных из регистра в прямом или обратном коде подается соответствующий управляющий сигнал.

Данные обычно выдаются из одного регистра и одновременно принимаются в другой (рис. 3.29). При этом используются регистры, построенные на синхронных *RS*-триггерах, а передача данных осуществляется в парафазном коде. В этом случае прием дан-

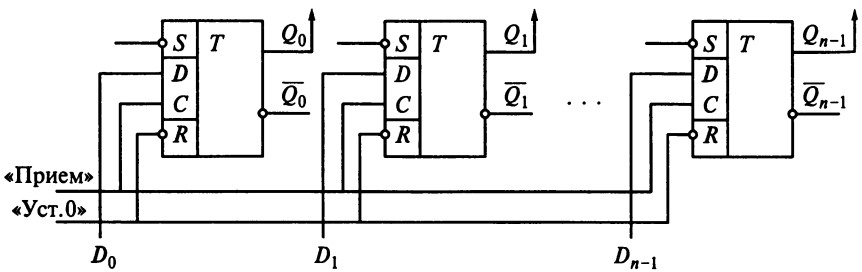


Рис. 3.27. Регистр с параллельной записью

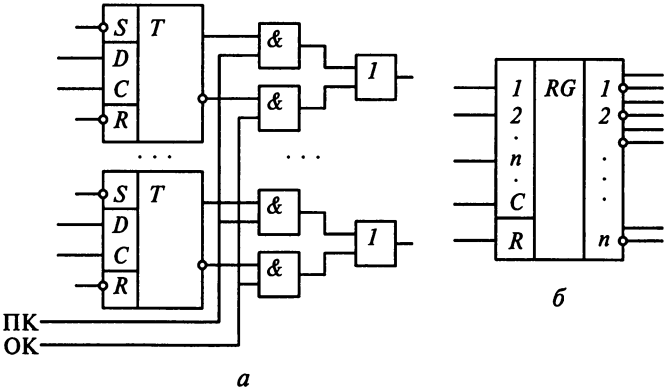


Рис. 3.28. Регистр:
а — схема выдачи данных из регистра; *б* — УГО

ных в регистр Рг1 производится при поступлении сигнала ПрРг1, а запись данных из Рг1 в регистр Рг2 — при поступлении сигнала ПрРг2.

Передача данных между регистрами может выполняться как прямо, т.е. без изменения номеров разрядов, так и со сдвигом влево или вправо на один или несколько разрядов.

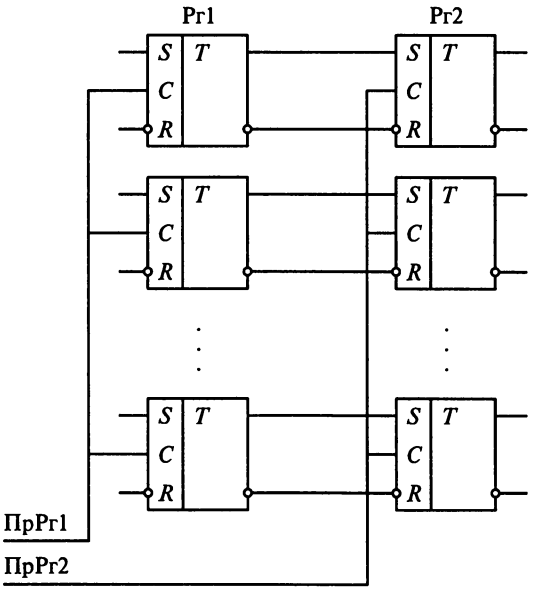


Рис. 3.29. Схема передачи данных из регистра в регистр

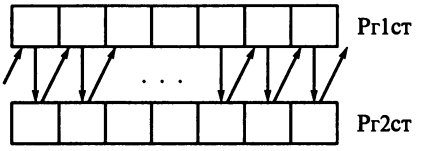


Рис. 3.30. Схема сдвига данных на один разряд вправо

Для сдвига информации в пределах одного регистра используются сдвигающие регистры (рис. 3.30). Сложность сдвига заключается в том, что каждый разряд регистра должен одновременно принять новую и передать старую информацию. Для устойчивой работы сдвигающего регистра в каждом разряде используют два однотактных триггера или один двухтактный, т.е. сдвигающий регистр фактически состоит из двух регистров.

Сдвигающий регистр представляет собой сдвоенный регистр и состоит из регистров первой (Рг1ст) и второй (Рг2ст) ступеней, между которыми производится обмен данными. Перед сдвигом в регистрах первой и второй ступеней записана одна и та же информация. При сдвиге данные сначала передаются из Рг2ст в Рг1ст со сдвигом вправо, затем возвращаются в Рг2ст без сдвига.

Схема регистра сдвига на один разряд вправо, выполненного на однотактных *RS*-триггерах, приведена на рис. 3.31. Сдвиг производится при поступлении сигнала «Сдвиг». Каждый сигнал «Сдвиг» вызывает сдвиг данных на один разряд. При этом в осво-

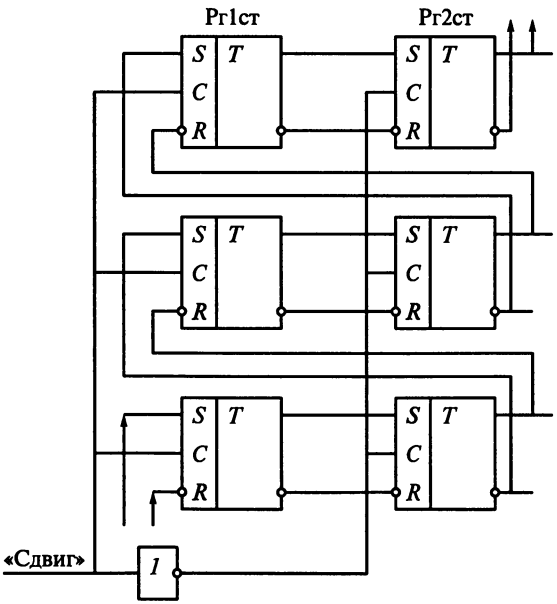


Рис. 3.31. Регистр сдвига вправо

бодившийся разряд записывается «0», а данные, выходящие из регистра, теряются.

Аналогичным образом можно построить регистр сдвига вправо, регистр со сдвигом данных на несколько разрядов в одном такте или реверсивный сдвигающий регистр, позволяющий сдвигать данные влево или вправо в зависимости от управляющего сигнала.

На основе сдвигающего регистра можно построить преобразователь последовательного кода в параллельный, и наоборот (рис. 3.32).

Преобразователь выполнен на основе регистра со сдвигом данных вправо на один разряд. Регистр сдвига состоит из двухтактных *JK*-триггеров с дополнительными асинхронными входами *S* и *R*. Для преобразования параллельного кода в последовательный регистр устанавливается в нулевое состояние сигналом «Уст. 0». Затем данные в параллельном коде поступают на дополнительные входы регистра *S*. При подаче сигнала «Сдвиг» данные в последовательном коде выдаются на последовательный выход регистра. Для преобразования последовательного кода в параллельный данные подаются на последовательный вход одновременно с поступлением сигналов «Сдвиг». После заполнения регистра данные могут быть выданы в параллельном коде с выходов D_1, D_2, ..., D_{n-1}.

Подача сигналов с выходов последнего триггера на входы *J* и *K* первого триггера позволяет получить кольцевой регистр сдвига, в котором разряды данных последовательно проходят через все триггеры, а состояние регистра периодически повторяется.

Счетчики. Счетчики предназначены для подсчета числа входных сигналов. Обычно входными сигналами являются импульсы напряжения. Результат счета — это число, представленное в дво-

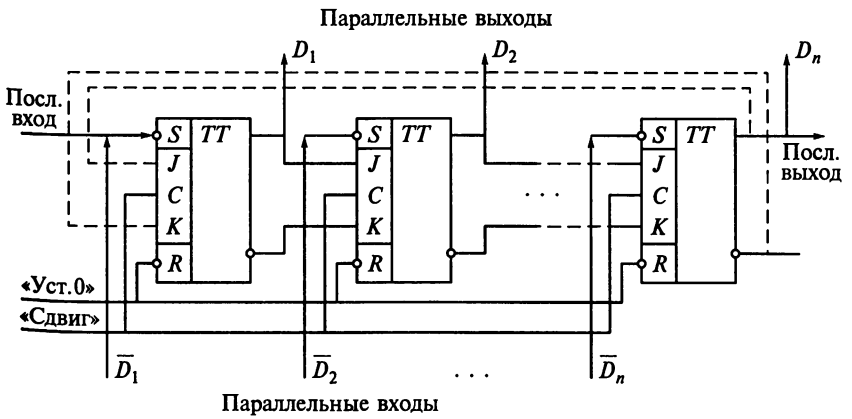


Рис. 3.32. Преобразователь параллельного кода в последовательный

ичной или двоично-десятичной системе счисления. В двоичном счетчике число разрядов зависит от модуля счета M (коэффициента пересчета), который определяет число состояний счетчика. При поступлении на вход счетчика M импульсов счетчик возвращается в исходное состояние, после этого состояния счетчика повторяются. Модуль счета на единицу больше максимального числа $K_{сч}$, которое может зафиксировать счетчик. В двоичных счетчиках с естественным порядком счета $M = 2^n$, где n — разрядность счетчика. Тогда $K_{сч} = 2^n - 1$. Такие счетчики имеют наиболее простую схему и строятся на T- или JK-триггерах.

В зависимости от направления счета счетчики могут быть суммирующими, вычитающими и реверсивными. *Суммирующие* счетчики при поступлении каждого входного импульса увеличивают показания на единицу, *вычитающие* — уменьшают. *Реверсивные* счетчики могут работать как в режиме суммирования, так и в режиме вычитания. Схемы трехразрядных двоичных счетчиков показаны на рис. 3.33, 3.34. Счетчики построены на двухтактных T-триггерах с дополнительными входами для установки счетчика в начальное состояние.

В суммирующем счетчике начальное (нулевое) состояние счетчика задается управляющим сигналом «Уст. 0» (см. рис. 3.33, *а*). Входные импульсы $U^+(t)$ поступают на вход триггера младшего разряда T. Этот триггер изменяет состояние по спаду каждого входного импульса, остальные триггеры меняют состояние по спаду сигнала на прямом выходе триггера соседнего младшего разряда. Таким образом, второй триггер меняет состояние после каждого второго импульса, третий — после каждого четвертого и т.д. При подаче на вход счетчика последовательности импульсов счетчик меняет состояния в соответствии с временно́й диаграммой (см. рис. 3.33, *б*). Так как триггер с выходом Q_1 фиксирует младший разряд результата счета, то состояния счетчика меняются в пос-

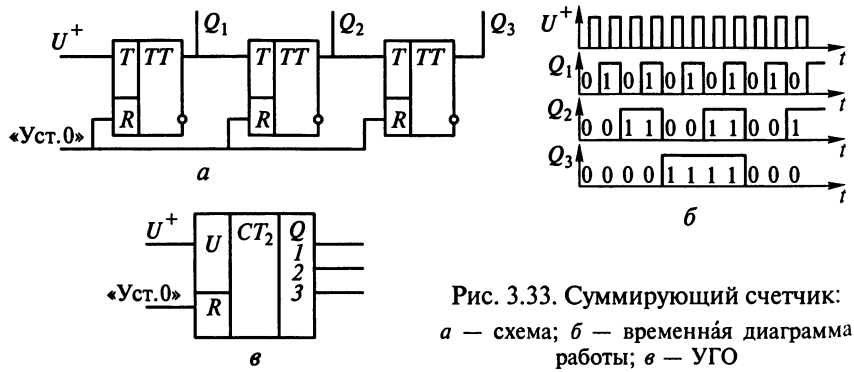
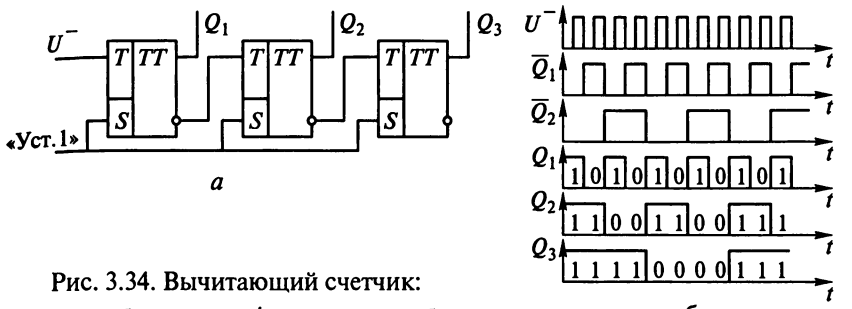


Рис. 3.33. Суммирующий счетчик:
а — схема; *б* — временна́я диаграмма работы; *в* — УГО



а *б*

Рис. 3.34. Вычитающий счетчик:
а — схема; *б* — временна́я диаграмма работы

ледовательности, соответствующей логике работы суммирующего счетчика по модулю 8:

$$Q_3Q_2Q_1 = 000 \rightarrow 001 \rightarrow 010 \rightarrow 011 \rightarrow 100 \rightarrow 101 \rightarrow 110 \rightarrow 111 \rightarrow \rightarrow 000 \ldots$$

Условное графическое обозначение суммирующего счетчика показано на рис. 3.33, *в*.

В вычитающих счетчиках на входы старших разрядов подаются сигналы с инверсных выходов триггеров младших разрядов. В начальном состоянии счетчика все триггеры установлены в состояние «1» сигналом «Уст. 1» (см. рис. 3.34, *а*). При поступлении входных импульсов первый триггер меняет состояние по спаду каждого входного импульса, а все остальные — по спаду сигнала на инверсном выходе триггера соседнего младшего разряда. Последовательность изменения состояний счетчика показана на временно́й диаграмме (см. рис. 3.34, *б*) и соответствует логике работы вычитающего счетчика по модулю 8:

$$Q_3Q_2Q_1 = 111 \rightarrow 110 \rightarrow 101 \rightarrow 100 \rightarrow 001 \rightarrow 010 \rightarrow 001 \rightarrow 000 \rightarrow \rightarrow 111 \ldots$$

В рассмотренных счетчиках входной сигнал (в худшем случае) должен последовательно пройти через все триггеры. Такие счетчики называются *счетчиками с последовательными переносами*. Для повышения быстродействия используют схемы со сквозными или параллельными переносами.

В счетчиках со сквозными переносами сигнал переноса последовательно проходит мимо каждого триггера, находящегося в состоянии «1», через вентиль на два входа. В таком счетчике время установления состояния пропорционально числу вентилей, т. е. разрядности счетчика. Схема суммирующего счетчика со сквозными переносами показана на рис. 3.35.

Счетчики с параллельными переносами используют вентили с большим числом входов. Это позволяет сигналам переноса прохо-

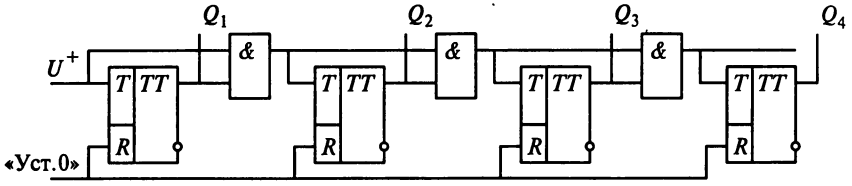


Рис. 3.35. Счетчик со сквозными переносами

дить мимо группы триггеров, находящихся в состоянии «1». Схема суммирующего счетчика с параллельными переносами показана на рис. 3.36.

Двоично-десятичные счетчики. Ранее рассмотренные счетчики являются счетчиками по модулю $M = 2^n$, где n — целое число ($M = 2, 4, 8, ...$). В некоторых схемах используют счетчики с модулем, отличным от $M = 2^n$. Например, в электронных часах используют счетчики по модулю 10 для подсчета единиц секунд, минут и часов, счетчики по модулю 6 для подсчета десятков секунд и минут, а также счетчики по модулю 2 или 3 для подсчета десятков часов.

Схемы счетчиков с произвольным модулем счета могут быть построены с применением теории автоматов как автоматы с памятью. Для построения таких счетчиков можно применить упрощенный метод. Его суть сводится к использованию известных схем счетчиков по модулю $M = 2^n$, где $M \geq M_{\text{треб}}$ ($M_{\text{треб}}$ — требуемое значение модуля счета). Схема счетчика дополняется схемой принудительного сброса счетчика в состояние «0» при поступлении на вход числа импульсов, равного требуемому значению модуля счета.

В частности, для построения счетчика по модулю 10 (двоично-десятичного счетчика) используется счетчик по модулю 16, т.е. четырехразрядный суммирующий двоичный счетчик. Такой счетчик может быть использован, например, в качестве счетчика единиц секунд электронных часов. В этом случае на его вход подаются импульсы частотой 1 Гц. Схема принудительного сброса счетчика

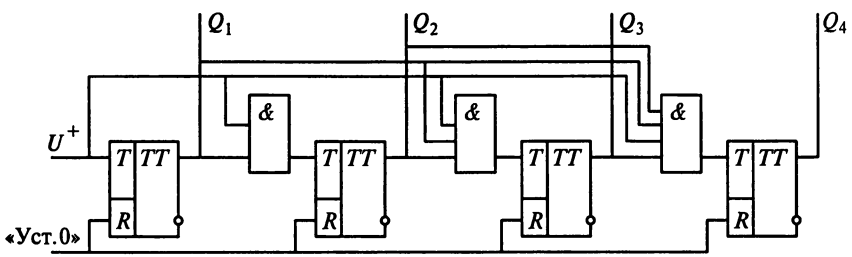


Рис. 3.36. Счетчик с параллельными переносами

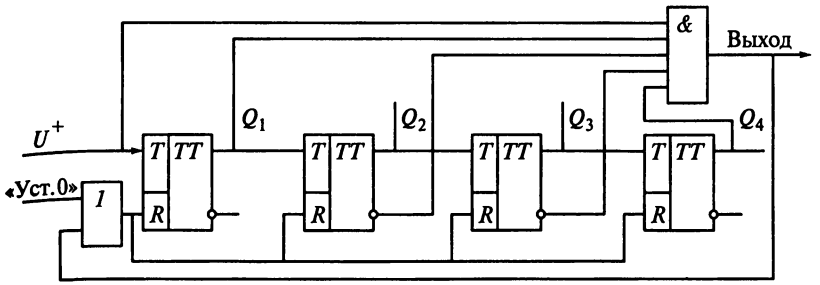


Рис. 3.37. Двоично-десятичный счетчик

в состояние «0» может быть выполнена в виде элемента И (рис. 3.37).

Исходное (нулевое) состояние счетчика устанавливается сигналом «Уст. 0». При поступлении входных импульсов U^+ счетчик начинает их подсчет как суммирующий двоичный счетчик. После девятого импульса счетчик переходит в состояние «1001». С приходом десятого импульса на всех входах элемента И устанавливаются единичные значения и сигнал с его выхода обнуляет счетчик. При поступлении следующих импульсов состояния счетчика меняются в соответствии с логикой работы счетчика по модулю 10 ($Q_4Q_3Q_2Q_1 = 0000 \rightarrow 0001 \rightarrow 1000 \rightarrow 1001 \rightarrow 0000 \rightarrow 0001...$).

Сигнал с выхода элемента И может быть использован как входной для последующей схемы (например, для счетчика десятков секунд). Такой счетчик выполняет функцию делителя частоты на 10.

Контрольные вопросы

1. Перечислите составные части компьютера в порядке уменьшения функциональной сложности.
2. Чем различаются узлы комбинационного и накапливающего типов?
3. Что такое логический элемент?
4. Что такое функционально полная система логических элементов?
5. Что такое состояние триггера?
6. Почему триггер имеет два устойчивых состояния?
7. Чем различаются асинхронный и синхронный триггеры?
8. Что такое таблица переходов триггера?
9. В каких случаях используются двухтактные триггеры?
10. Чем различаются триггеры разных типов?
11. Почему асинхронный *D*-триггер не используют в качестве элемента памяти?
12. Какие сигналы нужно подать на входы синхронного *RS*-триггера, чтобы записать в него «1»?
13. Что такое дешифратор?

14. Сколько входов и выходов имеет одноразрядный сумматор?

15. Чем различаются сумматоры с последовательными, сквозными и параллельными переносами?

16. Почему для сложения тетрад двоично-десятичных чисел нельзя использовать обычный двоичный сумматор?

17. Как получить ДК отрицательного двоично-десятичного числа?

18. Для чего используют мультиплексоры?

19. Из каких элементов состоит регистр?

20. Почему для сдвига данных используют двухтактные триггеры?

21. Поясните принцип работы преобразователя параллельного кода в последовательный.

22. В чем состоит отличие схем суммирующего и вычитающего счетчиков?

23. Поясните способ построения счетчиков с произвольным модулем.

ГЛАВА 4

АРИФМЕТИКО-ЛОГИЧЕСКОЕ УСТРОЙСТВО

4.1. Организация АЛУ

Арифметико-логическое устройство предназначено для выполнения арифметических и логических операций над данными различного формата. В зависимости от характера выполняемых операций и формата данных различают следующие типы операций в АЛУ:

арифметические операции над целыми числами с ФТ (операции целочисленной арифметики);

арифметические операции над числами с ПТ;

арифметические операции над двоично-десятичными числами (операции десятичной арифметики);

логические операции.

В зависимости от набора операций и особенностей реализации алгоритмов их выполнения организация АЛУ может быть различной.

По характеру использования аппаратных средств различают многофункциональные и блочные АЛУ. В *многофункциональных* АЛУ все операции выполняются с использованием в основном одних и тех же аппаратных средств, которые настраиваются на заданный тип операции. *Блочные* АЛУ включают в себя отдельные блоки для операций различного типа. Эти блоки могут работать параллельно, что повышает быстродействие АЛУ, однако требует больших затрат на оборудование.

Арифметико-логическое устройство можно рассматривать как композицию двух основных частей: операционного блока (ОБ) и блока управления (рис. 4.1).

Операционный блок выполняет преобразование данных, определяемое кодом операции. Блок управления организует работу операционного блока, формируя необходимые управляющие сигналы. На ОБ поступают операнды, а на его выходе формируется результат операции. В ходе выполнения операции ОБ выдает признаки (условия), характеризующие промежуточные и окончательные результаты. К таким признакам относятся, например, знаки операндов, равенство результата нулю, значение цифры множителя при умножении, знак остатка при делении и т.д. В зависимо-

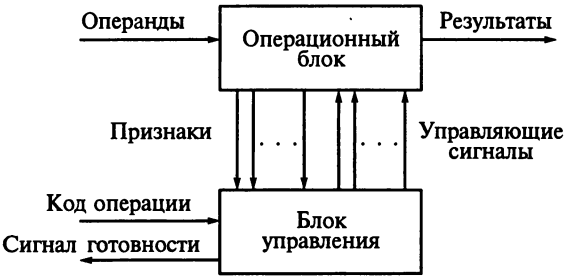


Рис. 4.1. Обобщенная структура АЛУ

сти от кода операции и значений признаков БУ выдает последовательность управляющих сигналов. После окончания операции он выдает сигнал готовности к выполнению следующей операции.

Операционный блок состоит из регистров и комбинационных схем. В зависимости от организации связей между регистрами и комбинационными схемами различают АЛУ с жесткой и магистральной структурами. В АЛУ с «*жесткой*» структурой комбинационные схемы закреплены за определенными регистрами (рис. 4.2). Между регистрами и комбинационными схемами существуют постоянные связи. Каждый регистр с закрепленными комбинационными схемами выполняет определенные микрооперации.

В состав ОБ АЛУ входят три регистра с закрепленными логическими схемами:

регистр первого слагаемого РСл1 со схемой ПРСл1;

регистр второго слагаемого РСл2 со схемой ПРСл2;

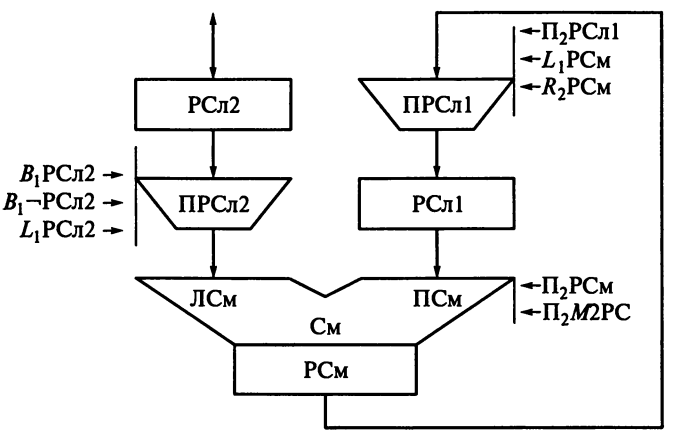


Рис. 4.2. Операционный блок АЛУ с «жесткой» структурой, выполняющий операции типа «сложение»

регистр суммы РСм со схемой комбинационного сумматора См.

Логическая схема ПРСл1 обеспечивает передачу результата из РСм в регистр РСл1:

в прямом коде РСл1:= РСм (по сигналу $П_2$РСл1);

со сдвигом влево на один разряд РСл1:= РСм(s*0)(по сигналу L_1РСм);

со сдвигом вправо на два разряда РСл1:= R_2(s*s*РСм) (по сигналу R_2РСм).

При сдвиге влево в освободившийся младший регистра РСл1 заносится «0». При сдвиге вправо в старшие разряды регистра заносится сигнал с левого входа сдвигателя, а содержимое младших разрядов регистра сумматора выдается на правый выход сдвигателя.

Логическая схема ПРСл2 обеспечивает передачу второго слагаемого на левый вход сумматора:

в прямом коде ЛСм:= РСл2 (по сигналу B_1РСл2);

в инверсном коде ЛСм:= ¬ РСл2 (по сигналу B_1¬РСл2);

со сдвигом влево на один разряд ЛСм:= L_1(РСл2*0) (по сигналу L_1РСл2).

Комбинационный сумматор выполняет суммирование (арифметическое или по модулю 2) операндов, поступивших на его входы (левый ЛСм и правый ПСм). Результат суммирования заносится в регистр сумматора РСм: РСм:=ЛСм + ПСм (по сигналу $П_2$РСм) или РСм:=ЛСм ⊕ ПСм (по сигналу $П_2M2$РСм).

В АЛУ с *магистральной* структурой (рис. 4.3) все регистры объединены в блок регистров общего назначения (РОН), а все комбинационные схемы составляют ОБ АЛУ. Обмен данными между регистрами и ОБ выполняется по магистралям, к которым может быть подключен любой из регистров или ОБ.

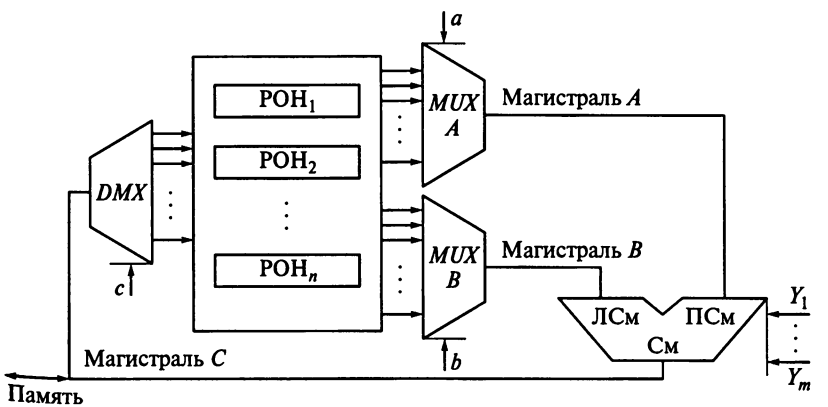


Рис. 4.3. Арифметико-логическое устройство магистральной структуры

Выходы каждого из регистров через мультиплексоры *A* и *B* могут быть подключены к входам *A* и *B* ОБ по магистралям *A* и *B*. Выбор подключаемых регистров производится по адресам *a* и *b*, которые поступают из устройства управления на адресные входы мультиплексоров. Операционный блок настраивается на выполнение заданной операции одним из сигналов управления Y_i ($i = 1, ..., m$), который поступает из устройства управления. Результат операции по магистрали *C* поступает на демультиплексор *DMX* и записывается в регистр блока РОН по адресу *c*.

В зависимости от числа магистралей различают трех-, двух- и одномагистральные АЛУ. Структура трехмагистрального АЛУ (рис. 4.4, *а*) позволяет выполнить за один такт чтение операндов из РОН, их суммирование в ОБ и запись суммы в РОН (рис. 4.4, *б*). Однако три магистрали занимают значительную часть кристалла микропроцессора.

Двухмагистральное АЛУ (рис. 4.5, *а*) содержит буферный регистр (БР) и требует для сложения двух машинных тактов. В первом такте по магистрали *B* из РОН в БР считывается операнд *B*. Во втором такте операнд *A* по той же магистрали считывается из РОН, операнды *A* и *B* суммируются в ОБ и результат *C* по магистрали *A* записывается в РОН (рис. 4.5, *б*).

Одномагистральное АЛУ использует два буферных регистра (рис. 4.6, *а*). В таком АЛУ сложение выполняется за три такта. В первом такте осуществляется чтение операнда *A* из РОН в $БР_1$, во втором — операнда *B* из РОН в $БР_2$, а в третьем суммируются операнды; результат записывается в один из регистров блока РОН (рис. 4.6, *б*).

Структура ОБ зависит от набора выполняемых операций и типа АЛУ (рис. 4.7). Основу ОБ составляет многофункциональный сумматор, который может выполнять микрооперации сложения (с уче-

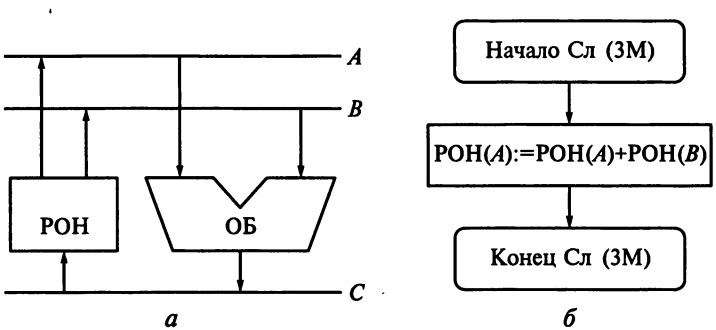


Рис. 4.4. Трехмагистральное АЛУ:
а — структура; *б* — микропрограмма сложения

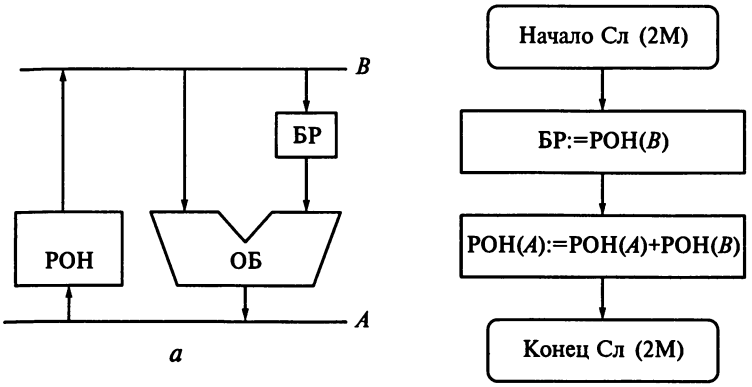


Рис. 4.5. Двухмагистральное АЛУ:
а — структура; *б* — микропрограмма сложения

том входящего переноса C_{IN}), сложение по модулю 2 (*М*2), логическое умножение И, а также логическое сложение ИЛИ:

Сл: См:= СмЛ + СмП + C_{IN};
*М*2: См:= СмЛ ⊕ СмП;
И: См:= СмЛ ∧ СмП;
ИЛИ: См:= СмЛ ∨ СмП.

Формирователи кодов $ФК_1$ и $ФК_2$ обеспечивают подачу на входы сумматора нулевых кодов, а также передачу сигналов с магистралей *A* и *B* прямым или инверсным кодом:

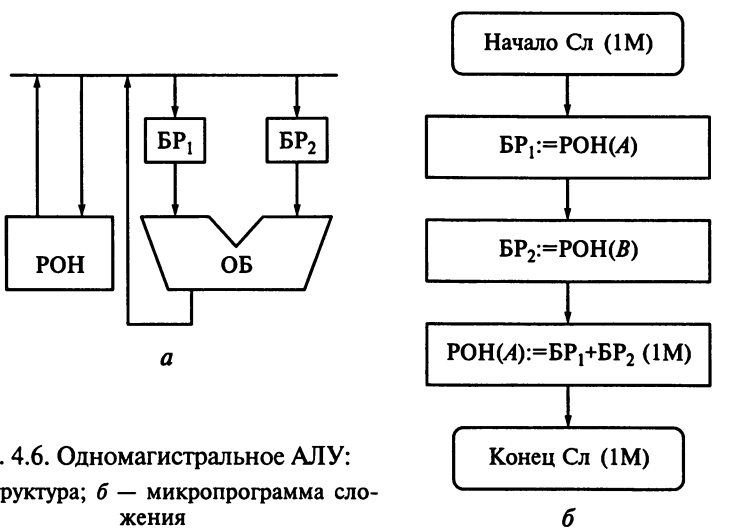


Рис. 4.6. Одномагистральное АЛУ:
а — структура; *б* — микропрограмма сложения

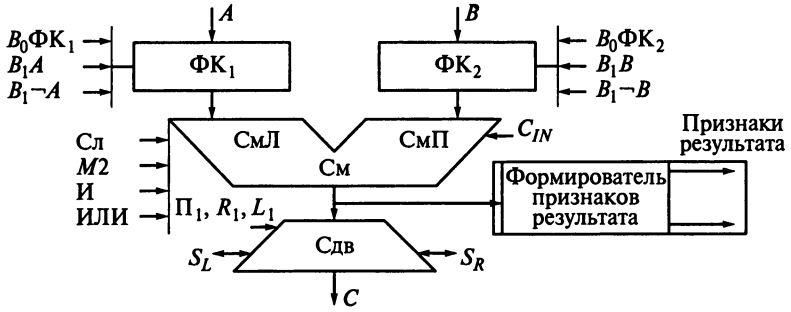


Рис. 4.7. Структура операционного блока магистрального АЛУ

B_0ФК$_1$: СмЛ:= 0; B_0ФК$_2$: СмП:= 0;
B_1A: СмЛ:= A; B_1B: СмП:= B;
$B_1\neg A$: СмЛ:= $\neg A$; $B_1\neg B$: СмП:= $\neg B$.

Сдвигатель позволяет выдавать результат из сумматора без сдвига, а также со сдвигом вправо или влево на один разряд:

$П_1$: C:= См;
R_1: C:= $R_1(S_L$ * См); S_R:= См(n);
L_1: C:= L_1(См * S_R); S_L:= См(0).

При сдвиге вправо на старший разряд магистрали C выдается сигнал с левого входа сдвигателя (S_L) и на правый выход сдвигателя выдается содержимое младшего разряда сумматора. При сдвиге влево на младший разряд магистрали C выдается сигнал с правого входа сдвигателя (S_R) и на левый выход сдвигателя содержимое старшего разряда сумматора.

Формирователь признаков результата (флагов) определяет значения признаков, которые используются для выполнения условных переходов. Такими признаками могут быть равенство результата нулю, знак результата, переполнение разрядной сетки и т.д.

4.2. Операции над числами с фиксированной точкой

Сложение и вычитание чисел с фиксированной точкой. Набор арифметических операций составляют операции сложения, вычитания, умножения и деления. Операция сложения является базовой, так как она входит в состав операций умножения и деления, а вычитание вообще выполняется как сложение с учетом знаков операндов (при использовании машинных кодов).

Сложение целых двоичных чисел с ФТ производится в следующем порядке:

1. Операнды представляются в обратном или дополнительном коде. Обычно используют ДК.

2. Сложение операндов производится поразрядно, начиная с младшего разряда.

3. В каждом разряде выполняются сложение двух цифр слагаемых и перенос из младшего разряда в соответствии с правилами двоичной арифметики.

4. Знаковые разряды участвуют в сложении наряду с цифровыми разрядами. Знак результата получается автоматически.

5. При сложении чисел в ДК перенос из знакового разряда отбрасывается. (При сложении чисел в ОК перенос из знакового разряда прибавляется к младшему разряду суммы.)

6. Если слагаемые представлены в ДК, то результат сложения получается также в ДК.

7. При сложении чисел с одинаковыми знаками возможно переполнение разрядной сетки.

Рассмотрим примеры.

Пример 4.1. Сложить числа $A_{10} = +12$ и $B_{10} = -7$ (сумма больше 0).

$A_2 = +1100 \rightarrow A_2^{ПК} = 0\ 1100 \rightarrow A_2^{ОК} = 0\ 1100 \rightarrow A_2^{ДК} = 0\ 1100$

$+ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad +$

$B_2 = -0111 \rightarrow B_2^{ПК} = 1\ 0111 \rightarrow B_2^{ОК} = 1\ 1000 \rightarrow B_2^{ДК} = 1\ 1001$

$A_2^{ДК} + B_2^{ДК} = C_2^{ДК} = \not{1}0\ 0101 \rightarrow C_2^{ПК} = 0\ 0101;$

$C_2 = +0101,\ C_{10} = +5.$

Пример 4.2. Сложить числа $A_{10} = -10$ и $B_{10} = +4$ (сумма меньше 0).

$A_2 = -1010 \rightarrow A_2^{ПК} = 1\ 1010 \rightarrow A_2^{ОК} = 1\ 0101 \rightarrow A_2^{ДК} = 1\ 0110$

$+ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad +$

$B_2 = +0100 \rightarrow B_2^{ПК} = 0\ 0100 \rightarrow B_2^{ОК} = 0\ 0100 \rightarrow B_2^{ДК} = 0\ 0100$

$A_2^{ДК} + B_2^{ДК} = C_2^{ДК} = 1\ 1010 \rightarrow C_2^{ПК} = 1\ 0110;$

$C_2 = -0110,\ C_{10} = -6.$

Пример 4.3. Сложить числа $A_{10} = -2$ и $B_{10} = -9$ с одинаковыми знаками (без переполнения).

$A_2 = -0010 \rightarrow A_2^{ПК} = 1\ 0010 \rightarrow A_2^{ОК} = 1\ 1101 \rightarrow A_2^{ДК} = 1\ 1110$

$+ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad +$

$B_2 = -1001 \rightarrow B_2^{ПК} = 1\ 1001 \rightarrow B_2^{ОК} = 1\ 0110 \rightarrow B_2^{ДК} = 1\ 0111$

$A_2^{ДК} + B_2^{ДК} = C_2^{ДК} = \not{1}1\ 0101 \rightarrow C_2^{ПК} = 1\ 1011;$

$C_2 = -1011,\ C_{10} = -11.$

При сложении чисел с одинаковыми знаками возможно получение результата, значение которого больше величины 2^{n-1}, где n — разрядность слагаемых. При этом результат не укладывается в отведенных для него разрядах, поэтому такая ситуация называется *переполнением разрядной сетки*.

Пример 4.4. Сложить числа $A_{10} = -13$ и $B_{10} = -6$ с одинаковыми знаками (с переполнением).

$A_2 = -1101 \rightarrow A_2^{\text{ПК}} = 1\ 1101 \rightarrow A_2^{\text{ОК}} = 1\ 0010 \rightarrow A_2^{\text{ДК}} = 1\ 0011$

$+$ $+$

$B_2 = -0110 \rightarrow B_2^{\text{ПК}} = 1\ 0110 \rightarrow B_2^{\text{ОК}} = 1\ 1001 \rightarrow B_2^{\text{ДК}} = 1\ 1010$

$A_2^{\text{ДК}} + B_2^{\text{ДК}} = C_2^{\text{ДК}} = \not{1}0\ 1101 \rightarrow C_2^{\text{ПК}} = 0\ 0101;$

$C_2 = +0101$. Знак? Переполнение!

Одним из признаков переполнения является отличие знака суммы от знака слагаемых. Иногда при сложении используется модифицированный дополнительный код (МДК), в котором для знака числа отводится два разряда. В этом случае признаком переполнения являются различные значения знаковых разрядов суммы. Если переполнение не возникает, то значения знаковых разрядов совпадают. Рассмотрим примеры сложения чисел в МДК.

Пример 4.5. Сложить числа $A_{10} = -12$ и $B_{10} = +5$ в МДК (сумма меньше 0).

$A_2 = -1100 \rightarrow A_2^{\text{ПК}} = 1\ 1100 \rightarrow A_2^{\text{ОК}} = 1\ 0011 \rightarrow A_2^{\text{МДК}} = 11\ 0100$

$+$ $+$

$B_2 = +0101 \rightarrow B_2^{\text{ПК}} = 0\ 0101 \rightarrow B_2^{\text{ОК}} = 0\ 0101 \rightarrow B_2^{\text{МДК}} = 00\ 0101$

$A_2^{\text{ДК}} + B_2^{\text{ДК}} = C_2^{\text{МДК}} = 11\ 1001 \rightarrow C_2^{\text{ПК}} = 1\ 0111;$

$C_2 = -0111$, $C_{10} = -7$.

Пример 4.6. Сложить числа $A_{10} = +11$ и $B_{10} = +8$ (сумма больше 0).

$A_2 = +1011 \rightarrow A_2^{\text{ПК}} = 0\ 1011 \rightarrow A_2^{\text{ОК}} = 0\ 1011 \rightarrow A_2^{\text{МДК}} = 00\ 1011$

$+$ $+$

$B_2 = +1000 \rightarrow B_2^{\text{ПК}} = 0\ 1000 \rightarrow B_2^{\text{ОК}} = 0\ 1000 \rightarrow B_2^{\text{МДК}} = 00\ 1000$

$A_2^{\text{ДК}} + B_2^{\text{ДК}} = C_2^{\text{МДК}} = 01\ 0011.$

Переполнение! Различные значения знаковых разрядов суммы.

Так как при сложении в дополнительном коде учитываются знаки чисел, то вычитание чисел можно заменить сложением в соответствии с соотношением

$$A - B = A + (-B).$$

Таким образом, вычитание заменяется сложением уменьшаемого с вычитаемым, взятым с обратным знаком.

Пример 4.7. Произвести вычитание чисел $A_{10} = +8$ и $B_{10} = -5$ (без переполнения).

$A_2 = +1000 \rightarrow A_2^{\text{ПК}} = 0\ 1000 \rightarrow \rightarrow \rightarrow \rightarrow A_2^{\text{ОК}} = 0\ 1000 \rightarrow A_2^{\text{ДК}} = 0\ 1000$

$+$

$$B_2 = -0101 \rightarrow B_2^{\text{ПК}} = 1\ 0101 \rightarrow (-B)_2^{\text{ПК}} = 0\ 0101 \rightarrow (-B)_2^{\text{ОК}} = 0\ 0101 \rightarrow$$
$$\rightarrow (-B)^{\text{ДК}}{}_2 = 0\ 0101$$

$$A_2^{\text{ДК}} + B_2^{\text{ДК}} = C_2^{\text{ДК}} = \pm 0\ 1101;$$

$C_2^{\text{ПК}} = 0\ 1101 \rightarrow C_2 = +1101 \rightarrow C_{10} = +13.$

При вычитании возможно переполнение, если числа имеют разные знаки.

Пример 4.8. Произвести вычитание чисел $A_{10} = -3$ и $B_{10} = +15$ (с переполнением).

$A_2 = -0011 \rightarrow A_2^{\text{ПК}} = 1\ 0011 \rightarrow A_2^{\text{ОК}} = 1\ 1100 \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow A_2^{\text{ДК}} = 1\ 1101$

$B_2 = +1111 \rightarrow B_2^{\text{ПК}} = 0\ 1111 \rightarrow (-B)_2^{\text{ПК}} = 1\ 111 \rightarrow (-B)_2^{\text{ОК}} = 1\ 0000 \rightarrow$
$\rightarrow (-B)_2^{\text{ДК}} = 1\ 0001$

$$A_2^{\text{ДК}} + B_2^{\text{ДК}} = C_2^{\text{ДК}} = \pm 0\ 1110;$$

$C_2^{\text{ПК}} = 0\ 1101 \rightarrow C_2 = +1101$? Переполнение!

Признаком переполнения в данном случае являются различные значения знаковых разрядов разности.

Выше рассматривались примеры сложения (вычитания) целых чисел с ФТ. Сложение дробных чисел с ФТ выполняется по тем же правилам.

Пример 4.9. Сложить дробные числа $A_{10} = -{}^{1}/_{16}$ и $B_{10} = +{}^{13}/_{16}$ с ФТ.

$A_2 = -0001 \rightarrow A_2^{\text{ПК}} = 1\ 0001 \rightarrow A_2^{\text{ОК}} = 1\ 1110 \rightarrow A_2^{\text{ДК}} = 1\ 1111$

$B_2 = +1101 \rightarrow B_2^{\text{ПК}} = 0\ 1101 \rightarrow B_2^{\text{ОК}} = 0\ 1101 \rightarrow B_2^{\text{ДК}} = 0\ 1101$

$$A_2^{\text{ДК}} + B_2^{\text{ДК}} = C_2^{\text{ДК}} = \pm 0\ 1100 \rightarrow C_2^{\text{ПК}} = 0\ 1100;$$

$C_2 = -1100,\ C_{10} = -{}^{12}/_{16}.$

Умножение чисел с фиксированной точкой. Умножение двоичных чисел проводится по существу так же, как и умножение десятичных. Умножение выполняется по шагам. Число шагов равно разрядности множителя. На каждом шаге множимое умножается на одну цифру множителя и получается *частичное произведение* (ЧП). Сумма всех ЧП, предварительно сдвинутых относительно друг друга, образует *полное произведение*. При этом направление сдвига зависит от используемого метода умножения. Разрядность произведения равна $2n$ при разрядности сомножителей, равной n.

Умножение целых и дробных чисел с ФТ выполняется по одинаковым алгоритмам. Отличие заключается только в том, что при умножении целых чисел для правильного расположения произве-

дения иногда в разрядной сетке необходим дополнительный сдвиг суммы ЧП вправо.

Так как цифры множителя могут принимать только значения «0» и «1», ЧП могут быть равны либо нулю, либо множимому, поэтому умножение двоичных чисел сводится к повторяющимся микрооперациям сложения и сдвига. Сложение ЧП производится в сумматоре с двумя входами, поэтому их сумма накапливается постепенно, на каждом шаге.

Различают следующие методы умножения (рис. 4.8):

1) начиная с младших разрядов множителя и сдвигом множимого влево;

2) начиная с младших разрядов множителя и сдвигом суммы ЧП вправо;

3) начиная со старших разрядов множителя и сдвигом множимого вправо;

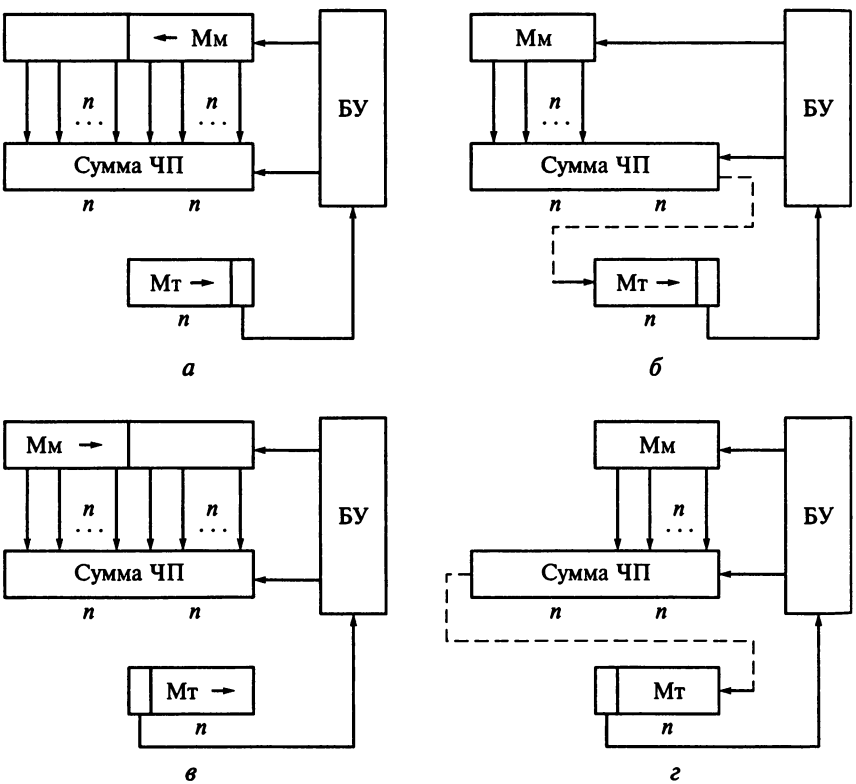


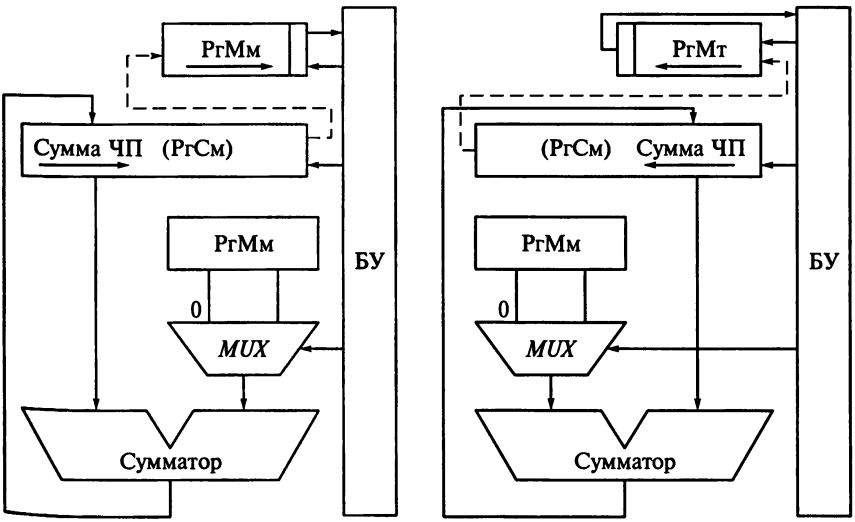
Рис. 4.8. Схемы реализации методов умножения:
а — метод 1; *б* — метод 2; *в* — метод 3; *г* — метод 4

4) начиная со старших разрядов множителя и сдвигом суммы ЧП влево.

Для реализации любого метода схема включает в себя регистр множимого Мм, регистр множителя Мт, сумматор с регистром сумматора для накопления суммы ЧП и блок управления БУ. При разрядности сомножителей, равной *n*, разрядность регистра множимого и сумматора зависит от метода умножения.

На каждом шаге умножения в БУ проводится анализ одной цифры множителя. В зависимости от ее значения БУ формирует сигналы для выдачи множимого в сумматор, если цифра множителя равна «1», а также сигналы сдвига на регистр множителя и множимого (или суммы ЧП).

Реализация методов 2 и 4 требует меньших аппаратурных затрат, так как при этом используется регистр множимого меньшей разрядности. Кроме того, при сдвиге множимого один из входов сумматора должен иметь удвоенную разрядность, что также увеличивает затраты на реализацию методов 1 и 3 (см. рис. 4.8, *а*, *в*). Анализ схем реализации методов 2 и 4 показывает, что затраты можно уменьшить, если использовать освобождающиеся разряды регистра множителя для записи уже сформированных разрядов произведения так, как это показано на рис. 4.8, *б*, *г* пунктирными линиями. При умножении используются методы со сдвигом ЧП (рис. 4.9).



а *б*

Рис. 4.9. Схемы умножения:

а — со сдвигом суммы ЧП вправо; *б* — со сдвигом суммы ЧП влево

Таблица 4.1

Умножение чисел с ФТ

Слагаемые, микрооперации	Метод умножения			
	1	2	3	4
Сумма ЧП	00000000	00000000	00000000	00000000
Сдвиг				←*
				00000000
Сложение	+	+	+	+
$ЧП_1$	00001010	10100000	01010000**	00001010
Сумма ЧП	= 00001010	= 10100000	= 01010000	= 00001010
Сдвиг		→		←
		01010000		00010100
Сложение	+	+	+	+
$ЧП_2$	00000000	00000000	00101000	00001010
Сумма ЧП	= 00001010	= 01010000	= 01111000	= 00011110
Сдвиг		→		←
		00101000		00111100
Сложение	+	+	+	+
$ЧП_3$	00101000	10100000	00000000	00000000
Сумма ЧП	= 00110010	= 11001000	= 01111000	= 00111100
Сдвиг		→		←
		01100100		01111000
Сложение	+	+	+	+
$ЧП_4$	01010000	10100000	00001010	00001010
Сумма ЧП	= 10000010	= 100000100***	= 10000010	= 10000010
Сдвиг		→		
Мм × Мт	10000010 (130_{10})	10000010 (130_{10})	10000010 (130_{10})	10000010 (130_{10})

* Фиктивный сдвиг, выполняется для регулярности умножения.
** Первое ЧП равно множимому, сдвинутому на один разряд вправо.
*** Возможно временное переполнение.

При умножении чисел со знаками отдельно определяются модуль произведения и его знак. Знак произведения определяется суммированием по модулю 2 знаков сомножителей. Вычисление модуля произведения может выполняться в прямом или дополнительном коде.

Пример определения модуля произведения различными методами в ПК при умножении чисел $Мм_{10} = 10$ ($Мм_2 = 1010$) и $Мт_{10} = 13$ ($Мт_2 = 1101$) приведен в табл. 4.1.

Умножение в ПК требует преобразования кодов, если числа хранятся в памяти в ДК, поэтому чаще умножение выполняется непосредственно в ДК. При этом произведение ДК не всегда равно ДК произведения. В зависимости от сочетания знаков сомножителей необходимо определенным образом корректировать результат. Далее приводятся значения поправок при умножении дробных чисел с ФТ.

1. Если $Мм > 0$ и $Мт > 0$, то $Мм^{ДК} = |Мм|$ и $Мт^{ДК} = |Мт|$.

Тогда $Мм^{ДК} \times Мт^{ДК} = |Мм| \times |Мт| = |Мм \times Мт| = (Мм \times Мт)^{ДК}$. Коррекция не нужна.

2. Если $Мм > 0$ и $Мт < 0$, то $Мм^{ДК} = |Мм|$ и $Мт^{ДК} = 1 - |Мт|$.

Тогда $Мм^{ДК} \times Мт^{ДК} = |Мм| \times (1 - |Мт|) = |Мм| - |Мм| \times |Мт|$.

Так как $(Мм \times Мт)^{ДК} = 1 - |Мм \times Мт| = 1 - |Мм| \times |Мт|$, то нужна коррекция и величина корректирующей поправки $\Delta = 1 - |Мм|$, т.е. дополнению множимого, взятого со знаком минус.

3. Если $Мм < 0$ и $Мт > 0$, то $Мм^{ДК} = 1 - |Мм|$ и $Мт^{ДК} = |Мт|$.

Тогда $Мм^{ДК} \times Мт^{ДК} = (1 - |Мм|) \times |Мт| = |Мт| - |Мм \times Мт|$.

Так как $(Мм \times Мт)^{ДК} = 1 - |Мм \times Мт| = 1 - |Мм| \times |Мт|$, то нужна коррекция и величина корректирующей поправки $\Delta = 1 - |Мт|$, т.е. дополнению множителя, взятого со знаком минус.

4. Если $Мм < 0$ и $Мт < 0$, то $Мм^{ДК} = 1 - |Мм|$ и $Мт^{ДК} = 1 - |Мт|$.

Тогда $Мм^{ДК} \times Мт^{ДК} = (1 - |Мм|)(1 - |Мт|) = 1 - |Мт| - |Мм| + |Мм \times Мт|$.

Так как $(Мм \times Мт)^{ДК} = |Мм \times Мт| = |Мм| \times |Мт|$, то нужна коррекция и величина корректирующей поправки $\Delta = |Мм| + |Мт| - 1$. Так как вычитание единицы приводит только к изменению знакового разряда, а знак определяется отдельно, то величина поправки может быть принята равной сумме модулей множимого и множителя, т.е. $\Delta = |Мм| + |Мт|$. Поправки вводятся на дополнительном шаге умножения.

Примеры умножения чисел в ДК по методу 2 приведены в табл. 4.2:

$|Мм| = 1010$; $(-|Мм|)^{ДК} = 0110$; $|Мт| = 1101$; $(-|Мт|)^{ДК} = 0011$.

Ускорение умножения. Умножение относится к длинным операциям, время выполнения которых значительно превышает время сложения. Для повышения быстродействия часто используют логические и аппаратные методы ускорения умножения. *Логические* методы основаны на одновременном анализе нескольких цифр множителя и сокращении за счет этого числа микроопераций. Ре-

Таблица 4.2

Умножение в дополнительном коде

Слагаемые, микрооперации	Знаки сомножителей			
	Мм > 0, Мт > 0	Мм > 0, Мт < 0	Мм < 0, Мт > 0	Мм < 0, Мт < 0
Сумма ЧП	00000000	00000000	00000000	00000000
Сложение	+	+	+	+
$ЧП_1$	10100000	10100000	01100000	01100000
Сумма ЧП	=10100000	=10100000	=01100000	=01100000
Сдвиг	→	→	→	→
	01010000	01010000	00110000	00110000
Сложение	+	+	+	+
$ЧП_2$	00000000	10100000	00000000	01100000
Сумма ЧП	=01010000	=11110000	=00110000	=10010000
Сдвиг	→	→	→	→
	00101000	01111000	00011000	01001000
Сложение	+	+	+	+
$ЧП_3$	10100000	00000000	01100000	00000000
Сумма ЧП	=11001000	=01111000	=01111000	=01001000
Сдвиг	→	→	→	→
	01100100	00111100	00111100	00100100
Сложение	+	+	+	+
$ЧП_4$	10100000	00000000	01100000	00000000
Сумма ЧП	=100000100	=00111100	=10011100	=00100100
Сдвиг	→	→	→	→
	10000010	00011110	01001110	00010010
Коррекция	Коррекция не нужна	+0110 Коррекция	+0011 Коррекция	+1010 +1101 Коррекция
Мм × Мт	10000010	=01111110	=01111110	=10000010
$(Мм \times Мт)^{дк}$	0 10000010	1 01111110	1 01111110	0 10000010

ализация логических методов приводит к усложнению БУ АЛУ. *Аппаратные* методы обеспечивают одновременное выполнение нескольких микроопераций за счет дополнительной аппаратуры. Часто используют комбинации логических и аппаратных методов.

Логические методы ускорения умножения. Самым простым методом ускорения является пропуск такта суммирования, если очередная цифра множителя равна нулю. В этом случае сокращается число микроопераций суммирования. Более эффективными являются методы с одновременным анализом нескольких цифр множителя. При этом уменьшается как число сложений, так и сдвигов.

В качестве примера рассмотрим модифицированный алгоритм Бута. Сущность метода заключается в том, что одновременно анализируются три разряда множителя: два текущих и старший разряд из предыдущей тройки. В зависимости от значения анализируемых разрядов выполняются действия, указанные в табл. 4.3.

Фактически в методе Бута на каждом шаге выполняется умножение множимого одновременно на две цифры множителя. Существуют методы умножения с непосредственной расшифровкой нескольких разрядов множителя. Наиболее распространен метод умножения с расшифровкой двух разрядов множителя (табл. 4.4).

Комбинация 11 = 100 – 1, поэтому прибавление утроенного множимого заменяется вычитанием множимого с прибавлением единицы к следующей паре разрядов. Вычитание множимого заменяется прибавлением дополнения множимого. При наличии переноса из предыдущей пары следующая пара увеличивается на единицу. На каждом шаге после суммирования сумма ЧП сдвига-

Таблица 4.3

Умножение по алгоритму Бута

Разряды множителя			Кратность множимому	Знак	Выполняемые действия
0	0	0	0	+	Не выполнять действий
0	0	1	1	+	Прибавить к сумме ЧП множимое
0	1	0	1	+	То же
0	1	1	2	+	Прибавить к сумме ЧП удвоенное множимое
1	0	0	2	–	Вычесть из суммы ЧП удвоенное множимое
1	0	1	1	–	Вычесть из суммы ЧП множимое
1	1	0	1	–	То же
1	1	1	0	–	Не выполнять действий

Таблица 4.4

Умножение с расшифровкой двух разрядов множителя

Значение пары разрядов	Перенос из предыдущей пары разрядов	Перенос в следующую пару разрядов	Знак частичного произведения	Кратность частичного произведения множимому
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

ется на два разряда вправо. Если возник перенос из последней пары, то на дополнительном шаге умножения к сумме ЧП добавляется множимое.

Пример 4.10. Выполнить умножение чисел, если $Мм_{10} = +27$, $Мт_{10} = -18$, $Мм_2 = +11011$, $Мт_2 = -10010$.

$Мм^{ПК} = 0\ 11011 \rightarrow Мм^{ДК} = 0\ 11011$. Модуль множимого $|Мм| = 11011$.

$Мт^{ПК} = 1\ 10010 \rightarrow Мт^{ДК} = 1\ 01110$. Модуль множителя $|Мт| = 10010$.

$(-Мм)^{ДК} = 1\ 00101$.

Последовательность формирования произведения с использованием метода Бута (модифицированного) и расшифровкой пар разрядов множителя показана в табл. 4.5 и 4.6 соответственно.

При использовании метода Бута умножение выполняется в МДК. При этом знак произведения формируется автоматически, так как знаки сомножителей участвуют в формировании ЧП. Произведение получается также в МДК.

При умножении с расшифровкой пар разрядов множителя сомножители представляются в ПК. Но вычитание в ходе умножения выполняют в МДК. Модуль произведения и его знак определяются отдельно. Для правильного расположения произведения в разрядной сетке выполняется дополнительный сдвиг суммы ЧП на один разряд вправо.

Аппаратные методы ускорения умножения. Аппаратные методы ускорения умножения реализуются с использованием матричных или древовидных схем.

Таблица 4.5

Метод Бута

Слагаемые, микрооперации	Множитель ($ЧП_i$)	Формирование суммы ЧП
Множитель	**1011100**	
Сумма ЧП		000000000000
Цифры Мт	**100**	+
$ЧП_1$	−2Мм	10 01010
Сумма ЧП		10 0101000000
Сдвиг		→→
		111001010000
Цифры Мт	**111**	+
$ЧП_2$	0	00 00000
Сумма ЧП		11 1001010000
Сдвиг		→→
		11 1110010100
Цифры Мт	**101**	+
$ЧП_3$	−Мм	11 00101
Сумма ЧП		11 0000110100
Дополнительный сдвиг		→
$(Мм \times Мт)^{МДК}$		11 1000011010
$(Мм \times Мт)^{ПК}$		1 0111100110
$(Мм \times Мт)_2$		−0111100110
$(Мм \times Мт)_{10}$		−486

Матричные умножители одновременно формируют разряды всех ЧП и их суммирование так, как это делается при умножении вручную «столбиком». Логика формирования разрядов ЧП и их суммирования при умножении четырехразрядных чисел без знака поясняется следующим примером:

$A = a_4a_3a_2a_1$; $B = b_4b_3b_2b_1$.

$$\begin{array}{cc} A & a_4a_3a_2a_1 \\ \times = C = & \times \\ B & b_4b_3b_2b_1 \end{array}$$

Таблица 4.6

Анализ пары разрядов

Слагаемые, микрооперации	Множитель ($ЧП_i$)	Формирование суммы ЧП
Множитель	**10010**	
Сумма ЧП		00 0000000000
Цифры Мт	**10**	+
$ЧП_1$	+2Мм	01 1011
Сумма ЧП		01 1011000000
Сдвиг		→→
		00 0110110000
Цифры Мт	**00**	+
$ЧП_2$	0	00 00000
Сумма ЧП		00 0110110000
Сдвиг		→→
		00 0001101100
Цифры Мт	**01**	+
$ЧП_3$	+Мм	00 11011
Сумма ЧП		00 1111001100
Дополнительный сдвиг		→
(|Мм|×|Мт|)		00 0111100110
$(Мм \times Мт)_2$		−0111100110
(Мм×Мт)10		−486

$$
\begin{array}{r}
a_4b_1\ a_3b_1\ a_2b_1\ a_1b_1 \\
a_4b_2\ a_3b_2\ a_2b_2\ a_1b_2 \\
a_4b_3\ a_3b_3\ a_2b_3\ a_1b_3 \\
a_4b_4\ a_3b_4\ a_2b_4\ a_1b_4 \\
\hline
A \times B = C
\end{array}
$$

Эта логика может быть реализована схемой, приведенной на рис. 4.10. Формирование разрядов ЧП вида a_ib_j выполняется логическими элементами И. В первом ярусе умножителя используются полусумматоры (*HS*), в следующих ярусах — одноразрядные сум-

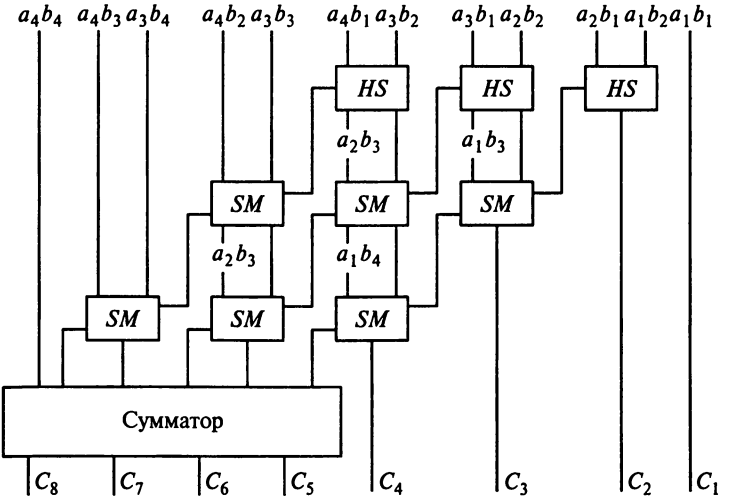


Рис. 4.10. Матричный умножитель

маторы (*SM*). Окончательное суммирование выполняется при помощи четырехразрядного сумматора, в котором возможна любая организация переносов. Числа со знаком при умножении могут представляться в ДК.

Время умножения в матричных умножителях пропорционально разрядности чисел. При большой разрядности чисел для уменьшения времени умножения применяются *древовидные* умножители. Повышение их быстродействия достигается за счет изменения организации суммирования ЧП по сравнению с матричными умножителями (рис. 4.11).

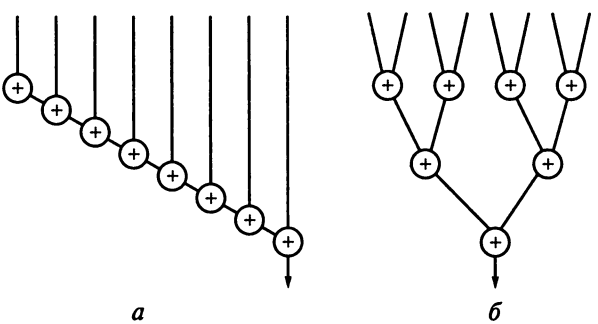


Рис. 4.11. Суммирование в умножителях:
а — с матричной структурой; *б* — с древовидной

Деление чисел с фиксированной точкой. Деление представляет собой более сложную операцию, чем умножение. При делении двоичных чисел используется тот же способ, что и при делении десятичных чисел вручную. Частное вычисляется по шагам. На первом шаге подбирается одна цифра частного. Для этого из делимого вычитается произведение делителя на подобранную цифру частного. Подбор производится с учетом знака и величины полученной разности (частичного остатка). На следующих шагах делитель вычитается из очередного частичного остатка. На каждом шаге производится сдвиг делителя вправо или остатка влево. В основном используется метод деления со сдвигом остатка влево. Деление продолжается до получения заданного числа разрядов частного.

Особенность деления двоичных чисел заключается в том, что цифры частного могут быть равными только нулю или единице, поэтому подбор отпадает и на каждом шаге вычитается только делитель.

При делении целых чисел делимое представляется обычно в формате двойного слова ($2n$ разрядов), делитель и частное имеют формат слова (n разрядов). При делении дробных чисел делимое может иметь формат слова.

Частное может содержать более чем n разрядов. В этом случае возникает переполнение разрядной сетки. Переполнение не возникает, если число, содержащееся в n старших разрядов делимого, меньше делителя (для целых чисел) или делимое меньше делителя (для дробных чисел). Одно из этих условий проверяется перед делением.

При делении чисел со знаком возможно отдельное определение модуля частного и его знака. В этом случае числа представляются в ПК. Возможно также выполнение деления непосредственно в ДК.

Известны два алгоритма деления: с восстановлением остатка и без его восстановления. Деление с восстановлением остатка выполняется в основном так же, как деление вручную. Общая последовательность деления с восстановлением остатка содержит следующие микрооперации:

1. Делитель размещается в старших n разрядах двойного слова.
2. Проверяется условие возможности деления (отсутствие переполнения). Для этого из делимого вычитается делитель и анализируется знак остатка.
3. Если остаток положительный, то деление невозможно, формируется признак переполнения и процесс заканчивается. Если остаток меньше нуля, то деление продолжается. При этом остаток восстанавливается путем прибавления делителя.
4. Остаток сдвигается влево на один разряд.
5. Из сдвинутого остатка вычитается делитель и анализируется знак остатка.

Делимое $A_{10}=+145$	00 1001 0001	1101	Делитель $B_{10}=-13$
Вычитание делителя	+ 11 0011	1011	Частное $(A/B)_{10}=-11$
Остаток < 0	= 11 1100 0001	↑↑↑↑	Деление корректно
Восстановление остатка	+ 00 1101	||||	
Восстановленный остаток	= 00 1001 0001	||||	
Сдвиг остатка влево	01 0010 0010	||||	
Вычитание делителя	+ 11 0011	||||	
Остаток > 0	= 00 0101 001	--↑|||	
Сдвиг остатка влево	00 1010 01	|||	
Вычитание делителя	+ 11 0011	|||	
Остаток < 0	= 11 1101 01	---↑||	
Восстановление остатка	+ 00 1101	||	
Восстановленный остаток	= 00 1010 01	||	
Сдвиг остатка влево	01 0100 1	||	
Вычитание делителя	+ 11 0011	||	
Остаток > 0	= 00 0111 1	-----↑|	
Сдвиг остатка влево	00 1111	|	
Вычитание делителя	+ 11 0011	|	
Остаток > 0	= 00 0010	------↑	Остаток $R_{10}=+2$

Рис. 4.12. Деление чисел с восстановлением остатка

6. Если остаток положительный, то очередная цифра частного равна единице. Если остаток меньше нуля, то очередная цифра множителя равна нулю. При этом остаток восстанавливается путем прибавления делителя.

7. Проверяется условие окончания деления. Если получены все цифры частного, то деление заканчивается, иначе выполняется переход к п. 4. Последний остаток восстанавливается, если он меньше нуля. Для этого к нему прибавляется делитель.

При вычитании делителя используется дополнительный или модифицированный дополнительный код.

Пример 4.11. Выполнить деление чисел с восстановлением остатка, если $A_{10} = +145$; $A_2 = +10010001$; $|A|^{ПК} = 0\ 10010001$; $|A|^{МДК} = 00\ 10010001$.

$B_{10} = -13$; $B_2 = -1101$; $|B|^{\text{ПК}} = 0\ 1101$;
$|B|^{\text{МДК}} = 00\ 1101$; $(-|B|)^{\text{МДК}} = 11\ 0011$.

Процесс определения цифр частного показан на рис. 4.12.

Если при делении с восстановлением остатка получен отрицательный остаток, то после восстановления остатка, сдвига восстановленного остатка и последующего вычитания делителя на шаге i будет получен следующий результат:

$$R_i = 2(R_{i-1} + B) - B = 2R_{i-1} + B,$$

где R_i — остаток на шаге i; R_{i-1} — остаток на шаге $i - 1$; B — делитель.

Множитель 2 возникает при сдвиге данных влево на один разряд.

Результат $2R_{i-1} + B$ может быть получен более простым путем, что и используется при делении без восстановления остатка. Такой вид деления отличается тем, что при получении отрицательного остатка он сдвигается влево и к нему прибавляется делитель для определения следующего остатка.

Делимое $A_{10} = +145$	00 1001 0001	1101	Делитель $B_{10} = -13$
Вычитание делителя	+ 11 0011	1011	Частное $(A/B)_{10} = -11$
Остаток < 0	= 11 1100 0001	↑↑↑↑	Деление корректно
Сдвиг остатка влево	11 1000 001	||||	
Прибавление делителя	+ 00 1101 001	||||	
Остаток > 0	= 00 0101 001	--↑|||	
Сдвиг остатка влево	00 1010 01	|||	
Вычитание делителя	+ 11 0011	|||	
Остаток < 0	= 11 1101 01	---↑||	
Сдвиг остатка влево	11 1010 1	||	
Прибавление делителя	+ 00 1101	||	
Остаток > 0	= 00 0111 1	----↑|	
Сдвиг остатка влево	00 1111	|	
Вычитание делителя	+ 11 0011	|	
Остаток > 0	= 00 0010	-----↑	Остаток $R_{10} = +2$

Рис. 4.13. Деление чисел без восстановления остатка

Пример 4.12. Выполнить деление чисел без восстановления остатка, если $A_{10} = +145$; $A_2 = +10010001$; $|A|^{ПК} = 0\ 10010001$; $|A|^{МДК} = 00\ 10010001$.
$B_{10} = -13$; $B_2 = -1101$; $|B|^{ПК} = 0\ 1101$;
$|B|^{МДК} = 00\ 1101$; $(-|B|)^{МДК} = 11\ 0011$.

Действия, выполняемые при делении, показаны на рис. 4.13.

Деление без восстановления остатка может быть реализовано устройством деления (рис. 4.14). При анализе его схемы можно установить, что в ее состав входят те же узлы, которые составляют схему умножения (см. рис. 4.9, *а*). Эти схемы отличаются лишь направлением сдвига содержимого регистра сумматора и регистра множителя (частного). Используя регистры со сдвигом в двух направлениях, можно построить комбинированную схему умножения-деления (рис. 4.15).

Младшие разряды $2n$-разрядного регистра сумматора используются в качестве регистра множителя-частного, что позволяет уменьшить аппаратные затраты.

Блок умножения-деления настраивается на выполнение заданной операции сигналом кода операции, поступающим из центрального устройства управления.

Ускорение деления. Деление, как и умножение, является длинной операцией, при этом время выполнения деления зависит от разрядности операндов. Для ускорения деления используются логические или аппаратные методы.

Логические методы ускорения деления предполагают анализ не только знака, но и нескольких разрядов остатка. Это позволяет в

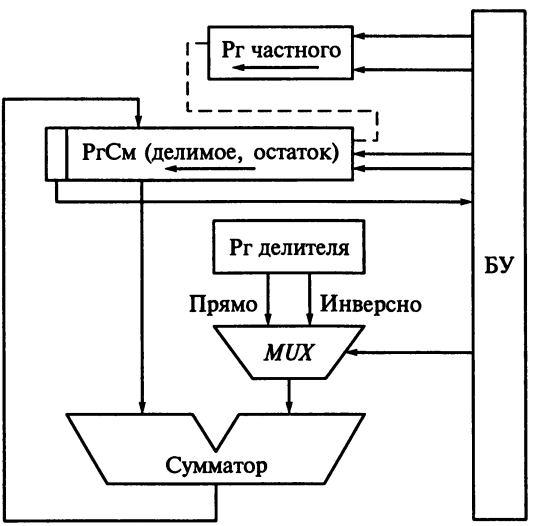


Рис. 4.14. Структура устройства деления

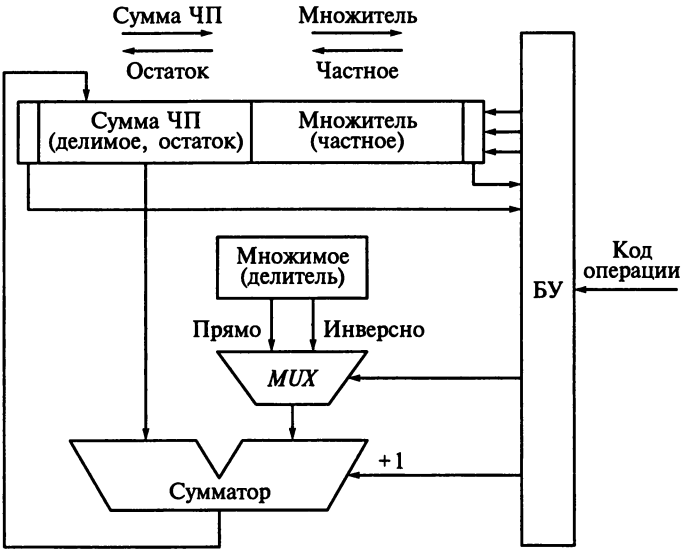


Рис. 4.15. Структура операционного блока умножения-деления

среднем сократить число микроопераций за счет некоторого усложнения БУ. Наиболее естественным является метод, который может быть использован при делении дробных чисел с ФТ, в частности мантисс чисел с ПТ. Так как мантиссы операндов перед делением нормализованы, то старший разряд мантиссы делителя равен единице. Если при делении с использованием модулей старший разряд остатка равен нулю, то остаток заведомо меньше делителя, поэтому на данном шаге можно принять очередную цифру частного равной нулю и выполнить сдвиг остатка влево без вычитания.

Признаком ускорения является наличие нуля по обе стороны точки.

Пример 4.13. Выполнить деление чисел, применяя логический метод ускорения с использованием МДК, если $A_{10} = -117$; $A_2 = -01110101$; $|A|^{ПК} = 0\ 01110101$; $|A|^{МДК} = 00\ 01110101$.

$B_{10} = +13$; $B_2 = +1101$; $|B|^{ПК} = 0\ 1101$; $|B|^{МДК} = 00\ 1101$; $(-|B|)^{МДК} = 11\ 0011$.

Процесс ускорения деления чисел с использованием МДК поясняется на рис. 4.16. Значения разрядов, при которых выполняется ускорение, выделены подчеркиванием.

В примере 4.13 ускорение можно выполнить только на шаге проверки корректности деления. Более эффективным является метод ускорения с анализом двух старших цифр остатка и делителя. На основе анализа возможных комбинаций их значений можно

Делимое $A_{10}=+117$	00 0111 0101	1101	Делитель $B_{10}=+13$
		1001	Частное $(A/B)_{10}=-9$
		↑↑↑↑	Деление корректно
Сдвиг остатка влево	00 1110 101	||||	Ускорение
Вычитание делителя	+ 11 0011	||||	
Остаток ≥ 0	= 00 0001 101	--↑|||	
Сдвиг остатка влево	00 0011 01	|||	
	+ 11 0011	|||	
Остаток < 0	= 11 0110 01	---↑||	
Сдвиг остатка влево	10 1100 1	||	
Прибавление делителя	+ 00 1101	||	
Остаток < 0	= 11 1001 1	----↑|	
Сдвиг остатка влево	11 0011	|	
Прибавление делителя	+ 00 1101	|	
Остаток ≥ 0	= 00 0000	-----↑	Остаток $R_{10}=0$

Рис. 4.16. Деление с ускорением

Делимое $A_{10}=+117$	00 0111 0101	1101	Делитель $B_{10}=+13$
		1001	Частное $(A/B)_{10}=-9$
		↑↑↑↑	Деление корректно
		||||	Ускорение
Сдвиг остатка влево	00 1110 101	||||	
Вычитание делителя	+ 11 0011	||||	
Остаток > 0	= 00 0001 101	--↑|||	
Сдвиг остатка влево	00 0011 01	---↑||	Ускорение
Сдвиг остатка влево	00 0110 1	----↑|	Ускорение
Сдвиг остатка влево	00 1101	|	
Вычитание делителя	+ 11 0011	|	
Остаток > 0	= 00 0000	-----↑	Остаток $R_{10}=0$

Рис. 4.17. Деление с анализом двух старших цифр остатка и делителя

сделать вывод, что остаток заведомо меньше делителя во всех случаях, кроме следующих комбинаций:

цифры остатка 10 11 11;

цифры делителя 10 10 11.

В остальных случаях возможно ускорение. При ускорении очередная цифра частного принимается равной нулю и выполняется сдвиг остатка влево на один разряд.

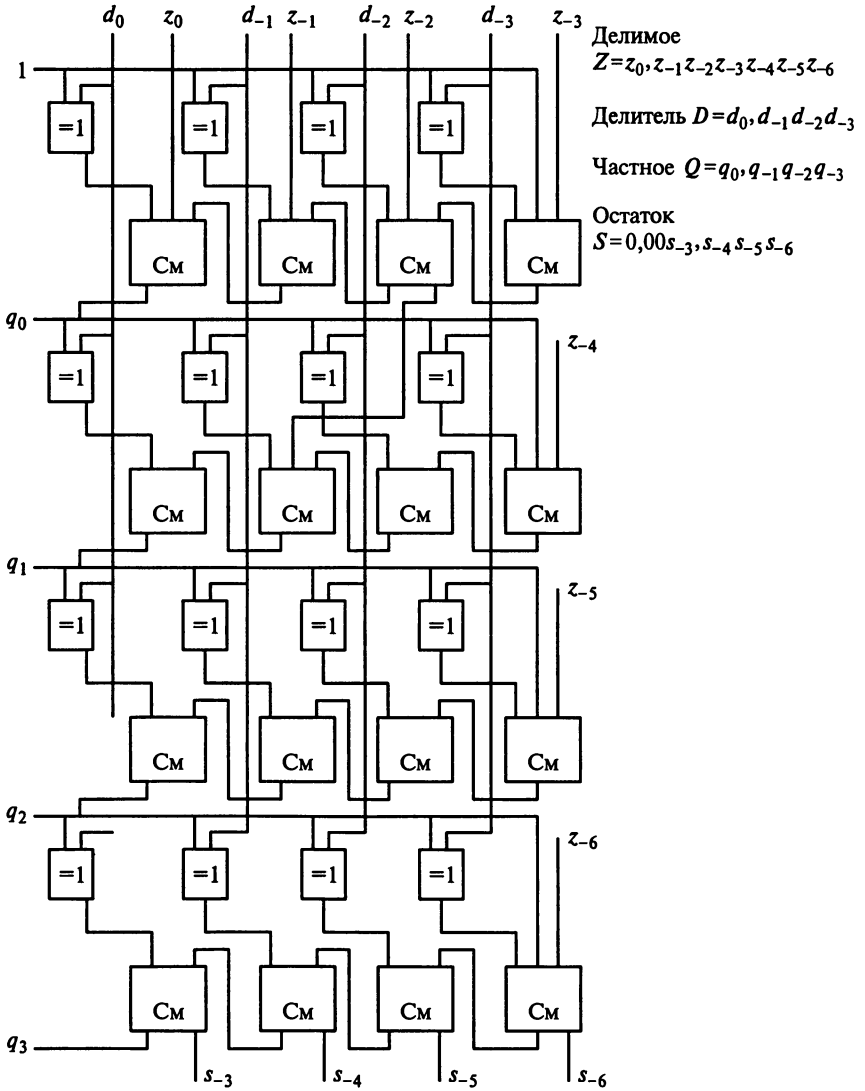


Рис. 4.18. Блок матричного деления

Пример 4.14. Выполнить деление чисел, применяя логический метод ускорения анализа старших цифр, если $A_{10} = -117$; $A_2 = -01110101$; $|A|^{ПК} = 0\ 01110101$; $|A|^{МДК} = 00\ 01110101$.

$B_{10} = +13$; $B_2 = +1101$; $|B|^{ПК} = 0\ 1101$; $|B|^{МДК} = 00\ 1101$; $(-|B|)^{МДК} = 11\ 0011$.

Последовательность деления показана на рис. 4.17, где выделены значения анализируемых разрядов, позволяющие выполнить ускорение.

Аппаратные методы ускорения деления в основном используют матричные схемы для ускоренного вычисления остатков (рис. 4.18). Эти схемы во многом аналогичны матричным схемам ускорения умножения.

Схема реализует алгоритм деления без восстановления остатка. Особенностью схемы является то, что в каждом ярусе сигнал переноса проходит через все сумматоры яруса, число которых равно разрядности делителя.

4.3. Операции с плавающей точкой

Операции с ПТ выполняются над числами, представленными в одинарном или двойном формате. Формат числа с ПТ включает в себя знак, мантиссу и порядок, над которыми выполняются разные действия. Поэтому операции над числами в форме с ПТ выполняются по более сложным алгоритмам, чем аналогичные операции над числами с ФТ. Операции над числами с ПТ можно разделить на три этапа: подготовительный, основной и заключительный.

Подготовительный этап включает в себя распаковку числа на три составляющие части и размещение их в регистрах операционного блока в зависимости от типа выполняемой операции, восстановление скрытой единицы в старшем разряде мантиссы, а также анализ операндов на равенство нулю.

В ходе *основного* этапа выполняется заданная операция и формируется ее результат.

Последовательность микроопераций основного этапа зависит от типа операции.

Заключительный этап включает в себя нормализацию результата и его округление, формирование признаков результата (флагов), а также упаковку составляющих результата в единый формат с удалением скрытой единицы. При операциях над числами с ПТ признак нулевого результата формируется как при получении нулевого значения мантиссы (потеря значимости мантиссы), так и при отрицательном переполнении порядка (потеря значимости порядка).

Сложение и вычитание чисел с ПТ. Особенностью сложения и вычитания чисел с ПТ является то, что в общем случае операнды могут иметь различные порядки. Суммировать разряды мантиссы можно только тогда, когда они имеют одинаковый вес, поэтому перед суммированием мантисс необходимо выровнять порядки. При этом могут быть потеряны младшие разряды мантиссы. Для уменьшения погрешности выполняется округление результата. После суммирования результат может оказаться ненормализованным, в этом случае необходима его нормализация.

С учетом этих особенностей сложение чисел с ПТ выполняется в следующем порядке: подготовительный этап, выравнивание порядков, сложение мантисс, заключительный этап.

На *подготовительном этапе* кроме распаковки операндов и восстановления скрытой единицы проводится анализ операндов на равенство нулю. Если один из них равен нулю, то результату присваивается значение другого операнда. На этом операция заканчивается.

Выравнивание порядков начинается с определения разности порядков $\Delta P = P_1 - P_2$, где P_1 — смещенный порядок первого числа; P_2 — смещенный порядок второго числа. Если разность порядков равна нулю, выравнивание порядков не производится. Если порядки не равны, то анализируется знак разности порядков и мантисса числа с меньшим порядком сдвигается вправо на величину разности порядков. При этом выдвигаемые младшие разряды мантиссы теряются (кроме старшего разряда, который используется при округлении). Сдвиг мантиссы выполняется за несколько тактов. В каждом такте мантисса сдвигается на один разряд, и модуль разности порядков уменьшается на единицу. Сдвиг продолжается до получения нулевой разности порядков. Порядок суммы принимается равным большему из порядков.

Сложение мантисс выполняется как сложение дробных чисел в формате с ФТ. При сложении обычно используется дополнительный или модифицированный дополнительный код.

На *заключительном этапе* выполняется нормализация результата. Если нормализация нарушена слева, выполняется сдвиг мантиссы суммы вправо на один разряд, при этом порядок суммы увеличивается на единицу. Если нормализация нарушена справа, производится сдвиг мантиссы влево на один разряд и проверяется значение признака нарушения нормализации. Если нормализация осталась нарушенной, сдвиг мантиссы повторяется. При каждом сдвиге порядок суммы уменьшается на единицу. В этом случае возможно отрицательное переполнение разрядной сетки порядка, поэтому формируется признак потери значимости порядка. Возможна также ситуация, когда во всех разрядах мантиссы записаны нули. При этом формируется признак потери значимости мантиссы.

После нормализации производится округление результата. Для этого к дополнительному разряду мантиссы прибавляется единица.

Рассмотрим пример сложения чисел с ПТ в ДК. Для упрощения операций с порядками используются несмещенные порядки чисел.

Пример 4.15. Выполнить сложение чисел с ПТ в ДК, если $A_{10} = -15^5/_8$; $A_2 = -1111.101$; $A_{\text{ПТ}} = -0.1111101 \times 10^{+100}$.

$B_{10} = +16^1/_4$; $B_2 = +10000{,}01$; $B_{\text{ПТ}} = +0.1000001 \times 10^{+101}$.

$(\text{M}_A)^{\text{ПК}} = 1\ 1111101$; $(\text{M}_A)^{\text{ДК}} = 1\ 0000011$; $(P_A)^{\text{ПК}} = 0\ 100$; $(P_A)^{\text{ДК}} = 0\ 100$; $(\text{M}_B)^{\text{ПК}} = 0\ 1000001$; $(\text{M}_B)^{\text{ДК}} = 0\ 1000001$; $(P_B)^{\text{ПК}} = 0\ 101$; $(P_B)^{\text{ДК}} = 0\ 101$; $(-P_B)^{\text{ДК}} = 1\ 011$.

Вычислим разность порядков:

$(P_A)^{\text{ДК}} = 0\ 100$

$+$

$(-P_B)^{\text{ДК}} = 1\ 011$

$(\Delta P)^{\text{ДК}} = 1\ 111 \rightarrow (\Delta P)^{\text{ПК}} = 1\ 001 \rightarrow (\Delta P)_2 = -001 \rightarrow (\Delta P)_{10} = -1$.

Разность порядков не равна нулю, выравнивание порядков необходимо. Разность порядков меньше нуля, число A имеет меньший порядок. Сдвигается мантисса числа A.

Мантисса числа A после сдвига: $(\text{M}_A)^{\text{ДК}} = 1\ 1000001\ |1$.

(В освобождающиеся разряды мантиссы заносится значение знакового разряда. Старший из выдвигаемых разрядов сохраняется в дополнительном разряде регистра суммы.)

Порядок суммы $(P_{A+B})^{\text{ДК}} = (P_B)^{\text{ДК}} = 0\ 101$.

Выполним сложение мантисс: $(\text{M}_A)^{\text{ДК}} = 1\ 1000001\ |1$

$+$

$(\text{M}_B)^{\text{ДК}} = 0\ 1000001\ |0$

$(\text{M}_A)^{\text{ДК}} + (\text{M}_B)^{\text{ДК}} = (\text{M}_A + \text{M}_B)^{\text{ДК}} = \overset{+}{-}0\ 0000010\ |1$. Нормализация нарушена справа.

Мантисса суммы сдвигается влево на пять разрядов.

Мантисса суммы после сдвига: $(\text{M}_A + \text{M}_B)^{\text{ДК}} = 0\ 1010000\ |0$

Округление. $+\qquad\qquad |1$

$=$_____

Мантисса суммы после округления: $(\text{M}_A + \text{M}_B)^{\text{ДК}} = 0\ 1010000$.

Порядок суммы после нормализации: $(P_{A+B})^{\text{ДК}} = 0\ 101 + 1\ 011 = 0\ 000$.

Результат: $(\text{M}_A + \text{M}_B)^{\text{ПК}} = 0\ 1010000 \rightarrow (\text{M}_A + \text{M}_B)_2 = +0.1010000$.

$(P_{A+B})^{\text{ДК}} = 0\ 000 \rightarrow (P_{A+B})_2 = +000$.

Сумма: $(A + B)_{\text{ПК}} = +0.1010000 \times 10^{+000} \rightarrow (A + B)_2 = +0.101 \rightarrow (A + B)_{10} = +5/8$.

Точный результат: $-15^5/_8 + 16^1/_4 = +^5/_8$.

При выполнении операции вычитания знак вычитаемого изменяется на противоположный, далее операция продолжается как сложение.

Пример 4.16. Определить разность чисел, представленных в форме с ПТ. Операцию выполнить с использованием МДК.

$A_{10} = -5^3/_4$; $A_2 = -101.11$; $A_{\text{ПТ}} = -0.10111 \times 10^{+011}$.

$B_{10} = +3\ ^5/_8$; $B_2 = +11.101$; $B_{\text{ПТ}} = +0.11101 \times 10^{+010}$.

$(\text{M}_A)^{\text{ПК}} = 1\ 10111$; $(\text{M}_A)^{\text{МДК}} = 11\ 01001$; $(P_A)^{\text{ПК}} = 00\ 011$; $(P_A)^{\text{МДК}} = 00\ 011$;

$(\text{M}_B)^{\text{ПК}} = 0\ 11101$; $(\text{M}_B)^{\text{МДК}} = 00\ 11101$; $(-\text{M}_B)^{\text{МДК}} = 11\ 00011$.

$(P_B)^{\text{ПК}} = 00\ 010$; $(P_B)^{\text{МДК}} = 00\ 010$; $(-P_B)^{\text{МДК}} = 11\ 110$.

Вычислим разность порядков:

$(P_A)^{\text{МДК}} = 00\ 011$

$+$

$(-P_B)^{\text{МДК}} = 11\ 110$

$(\Delta P)^{\text{МДК}} = 00\ 001 \rightarrow (\Delta P)^{\text{ПК}} = 00\ 001 \rightarrow (\Delta P)_2 = +001 \rightarrow (\Delta P)_{10} = +1$.

Разность порядков не равна нулю, выравнивание порядков необходимо. Разность порядков больше нуля, число B имеет меньший порядок. Сдвигается мантисса числа B.

Мантисса числа $(-B)$ после сдвига: $(-\text{M}_B)^{\text{МДК}} = 11\ 10001|1$.

Порядок разности $(P_{A-B})^{\text{МДК}} = (P_A)^{\text{МДК}} = 00\ 011$.

Выполним сложение мантисс: $(\text{M}_A)^{\text{МДК}} = 11\ 01001|0$

$+$

$(-\text{M}_B)^{\text{МДК}} = 11\ 10001|1$

$(\text{M}_A)^{\text{МДК}} + (-\text{M}_B)^{\text{МДК}} = (\text{M}_A - \text{M}_B)^{\text{МДК}} = \ \ddagger 10\ 11010|1$. Нормализация нарушена слева.

Мантисса разности сдвигается вправо на один разряд.

Мантисса разности после сдвига: $(\text{M}_A - \text{M}_B)^{\text{МДК}} = 11\ 01101|0$

Округление. $+ \qquad\qquad |1$

Мантисса разности после округления: $(\text{M}_A - \text{M}_B)^{\text{МДК}} = 11\ 01101$.

Порядок разности после нормализации: $(P_{A-B})^{\text{МДК}} = 00\ 011 + 00\ 001 = 00\ 100$.

Результат: $(\text{M}_A - \text{M}_B)^{\text{ПК}} = 11\ 10011 \rightarrow (\text{M}_A - \text{M}_B)_2 = -0.10011$.

$(P_{A-B})^{\text{МДК}} = 0\ 100 \rightarrow (P_{A-B})_2 = +100$.

$(A - B)_{\text{ПТ}} = -0.10011 \times 10^{+100} \rightarrow (A - B)_2 = -1001.1 \rightarrow (A - B)_{10} = -9^1/_2$.

Точный результат: $-5^3/_4 - 3^5/_8 = -9^3/_8$.

Погрешность возникла при сдвиге мантиссы числа B при выравнивании порядков.

Умножение чисел с ПТ. Произведение чисел в этом случае может быть определено следующим образом:

$$(A)_{\text{ПТ}} \times (B)_{\text{ПТ}} = (\text{M}_A \times 2^{P_A})(\text{M}_B \times 2^{P_B}) = (\text{M}_A \times \text{M}_B)(2^{P_A} + 2^{P_B}).$$

Таким образом, при умножении чисел с ПТ мантисса произведения равна произведению мантисс сомножителей, а порядок произведения — сумме их порядков. Умножение мантисс производится как умножение дробных чисел с ФТ, а сложение поряд-

ков — как сложение целых чисел с ФТ. Если порядки смещены, то при сложении порядков сумму порядков следует уменьшить на величину смещения.

Умножение чисел с ПТ выполняется в следующем порядке: подготовительный этап, сложение порядков, умножение мантисс, заключительный этап.

Если на подготовительном этапе один из операндов окажется равным нулю, то произведение принимается равным нулю и операция на этом заканчивается. Если произведение не равно нулю, выполняется сложение порядков. При использовании смещенных порядков перед суммированием один из порядков уменьшается на величину смещения, чтобы избежать переполнения порядков. При суммировании порядков в зависимости от их величины возможно переполнение порядка или потеря его значимости и формирование соответствующего признака. В этом случае операция заканчивается и ОС выдает соответствующее сообщение.

Если порядок произведения имеет допустимую величину, выполняется умножение мантисс сомножителей. Мантиссы перемножаются как дробные числа с ФТ. Результат умножения должен иметь одинарную разрядность, поэтому младшие разряды произведения отбрасываются (старший из них остается в дополнительном разряде).

На заключительном этапе проводится нормализация результата. При этом возможна потеря цифровых разрядов. Для уменьшения погрешности проводится округление путем прибавления единицы к дополнительному разряду. Результат упаковывается и отправляется в память.

Пример 4.17. Выполнить умножение чисел в формате с ПТ.

$A_{10} = -2^5/_8$; $A_2 = -10.101$; $A_{\text{ПТ}} = -0.10101 \times 10^{+010}$.

$B_{10} = +5^1/_4$; $B_2 = +101.01$; $B_{\text{ПТ}} = +0.10101 \times 10^{+011}$.

$(\text{M}_A)^{\text{ПК}} = 1\ 10101$; $(\text{M}_A)^{\text{ДК}} = 1\ 01011$; $(P_A)^{\text{ПК}} = 0\ 010$; $(P_A)^{\text{ДК}} = 0\ 010$;

$(\text{M}_B)^{\text{ПК}} = 0\ 10101$; $(\text{M}_B)^{\text{ДК}} = 0\ 10101$; $(P_B)^{\text{ПК}} = 0\ 010$; $(P_B)^{\text{ДК}} = 0\ 011$.

Вычислим сумму порядков: $(P_A)^{\text{ДК}} + (P_B)^{\text{ДК}} = 0\ 011 + 0\ 010 = 0\ 101$. (Сложение выполняется с использованием несмещенных порядков.)

Умножение модулей мантисс выполняем начиная со старших разрядов множителя и сдвигом суммы ЧП влево (табл. 4.7):

$(|\text{M}_A|^{\text{ПК}} \times |\text{M}_B|^{\text{ПК}}) = |\text{M}_A \times \text{M}_B|^{\text{ПК}} = 0.0110111001$.

Порядок произведения: $(P_{A\times B}) = (P_A)^{\text{ДК}} + (P_B)^{\text{ДК}} = 0\ 101$. $(P_{A\times B})_2 = +101$.

Нормализация нарушена справа.

Модуль мантиссы произведения после нормализации:

$|\text{M}_A \times \text{M}_B| = 0.\ 1101110010$.

Порядок произведения после нормализации: $(P_{A\times B})_2 = +\ 100$.

Модуль мантиссы произведения после округления:

$|\text{M}_A \times \text{M}_B|^{\text{ПК}} = 0.11011 \mid 10010.$

$+ \qquad\qquad \mid 1$

$|\text{M}_A \times \text{M}_B|^{\text{ПК}} = 0.11100.$

Таблица 4.7

Формирование модуля мантиссы

Значение очередного разряда множителя	Микрооперации	Частичные произведения и их сумма
	Исходное состояние	0000000000
	Сдвиг	←
	Сумма ЧП	0000000000
1	Сложение	+
	$ЧП_1$	10101
	Сумма ЧП	= 0000010101
	Сдвиг	←
	Сумма ЧП после сдвига	0000101010
0	$ЧП_2 = 0$; сдвиг	←
	Сумма ЧП после сдвига	0001010100
1	Сложение	+
	$ЧП_3$	10101
	Сумма ЧП	0001101001
	Сдвиг	←
	Сумма ЧП после сдвига	0011010010
0	$ЧП_4 = 0$; сдвиг	←
	Сумма ЧП после сдвига	0110100100
1	Сложение	+
	$ЧП_5$	10101
	Сумма ЧП	0110111001

Результат: $(A \times B)_2 = -0.11100 \times 10^{+100} \rightarrow (A \times B)_{10} = (-{}^{7}/_{8}) \times 2^{+4} = -14$.
Точный результат: $(-2^{5}/_{8}) \times 5^{1}/_{4} = 13^{25}/_{32}$.
Погрешность возникла при отбрасывании младших разрядов произведения.

Деление чисел с ПТ. При делении чисел с ПТ частное определяется следующим образом:

$$(A)_{ПТ}/(B)_{ПТ} = (M_A \times 2^{P_A})/(M_B \times 2^{P_B}) = (M_A/M_B)(2^{P_A} - 2^{P_B}).$$

Мантисса частного равна частному от деления мантисс, а порядок частного — разности порядков делимого и делителя. Деление мантисс выполняется как деление дробных чисел с ФТ, а вычитание порядков — как вычитание целых чисел с ФТ.

Пример 4.18. Выполнить деление чисел с ПТ, если $A_{10} = -4^1/_2$; $A_2 = -100.1$; $A_{\text{ПТ}} = -0.1001 \times 10^{+011}$.

$B_{10} = +1^5/_8$; $B_2 = +110.1$; $B_{\text{ПТ}} = +0.1101 \times 10^{+001}$.

$(\text{M}_A)^{\text{ПК}} = 1\ 1001$; $(\text{M}_A)^{\text{ДК}} = 1\ 0111$; $(P_A)^{\text{ПК}} = 0\ 011$; $(P_A)^{\text{ДК}} = 0\ 011$.

$(\text{M}_B)^{\text{ПК}} = 0\ 1101$; $(\text{M}_B)^{\text{ДК}} = 0\ 1101$; $(P_B)^{\text{ПК}} = 0\ 001$; $(P_B)^{\text{ДК}} = 0\ 001$.

$(-P_B)^{\text{ДК}} = 1\ 111$.

Вычислим разность порядков: $(P_A)^{\text{ДК}} - (P_B)^{\text{ДК}} = 0.011 + 1.111 = 0\ 010$. Вычитание производим с использованием несмещенных порядков. Порядок частного $P_{A/B} = +10$.

Делимое $A_{10} = -4^1/_2$	00 1001 0000	1101	Делитель $B_{10} = +1^5/_8$
Вычитание делителя	+ 11 0011	10110	Частное
			$(A/B)_{10} = -2^3/_4$
Остаток < 0	= 11 1100 0000	↑↑↑↑↑	Деление корректно
Сдвиг остатка влево	11 1000 000	|||||	
Прибавление делителя	+ 00 1101	|||||	
Остаток > 0	= 00 0101 000	--↑||||	
Сдвиг остатка влево	00 1010 00	||||	
Вычитание делителя	+ 11 0011	||||	
Остаток < 0	= 11 1101 00	---↑|||	
Сдвиг остатка влево	11 1010 0	|||	
Прибавление делителя	+ 00 1101	|||	
Остаток > 0	= 00 0111	-----↑||	
Сдвиг остатка влево	00 1110	||	
Вычитание делителя	+ 11 0011	||	
Остаток > 0	= 00 0001	------↑|	Остаток $R_{10} = +1/_8$
Сдвиг остатка влево	00 0010	|	
Вычитание делителя	11 0011	|	
Остаток < 0	11 0101	-------↑	Дополнительный разряд для округления

Рис. 4.19. Вычисление модуля мантиссы частного

Вычисление модуля мантиссы частного выполняется путем деления модуля мантиссы делимого на модуль мантиссы делителя (рис. 4.19). При делении используется МДК:

$(|M_A|)^{ДК} = 00.1001$; $(|M_B|)^{ДК} = 00.1101$; $(-|M_B|)^{ДК} = 11.0011$.

Модуль мантиссы частного:

$(|M_A|/|M_B|)^{ПК} = (|M_A/M_B|)^{ПК} = 0.1011 \mid 0$.

Порядок частного: $(P_A/_B) = +10$. Нормализация не нарушена.

Модуль мантиссы частного после округления:

$|M_A/M_B|^{ПК} = 0.1011 \mid 0$

$+ \qquad\quad \mid 1$

$|M_A/M_B|^{ПК} = 0.1011$.

Результат: $(A/B)_2 = -0.1011 \times 10^{+10} \rightarrow (A/B)_{10} = (-{}^{11}/_{16}) \times 2^{+2} = -2^3/_4$.

Точный результат: $(-4^1/_2)/1^5/_8 = -{}^{36}/_{13} \approx -2{,}77$. Погрешность возникла за счет отбрасывания младших разрядов частного.

Логические операции. Такие операции выполняются над *логическими данными*, которые представляют собой совокупность не зависимых друг от друга битов (разрядов). Операции производятся над каждым битом отдельно, результаты операции над отдельными битами никак не связаны друг с другом. К основным логическим операциям обычно относят операции отрицания (инверсии), логического сложения (дизъюнкции), логического умножения (конъюнкции), а также сложения по модулю 2. Логические операции могут быть реализованы либо с помощью многофункционального сумматора, либо специализированным блоком логических операций.

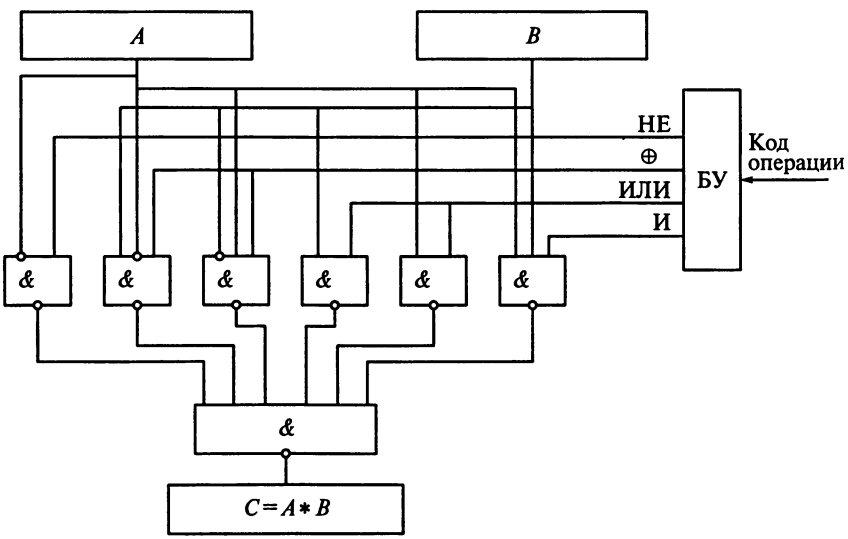


Рис. 4.20. Структура операционного блока логических операций

Функциональная схема ОБ логических операций представляет собой набор одноразрядных схем (рис. 4.20). Разряды операндов *A* и *B* поступают на комбинационную схему, выполненную на элементах И—НЕ с прямыми и инверсными входами. Блок управления в соответствии с кодом операции выдает управляющие сигналы и задает режим работы схемы для выполнения заданной операции. На выходе комбинационной схемы формируется результат $C = A * B$, где символ «*» означает заданную логическую операцию. Кроме рассмотренных выше операций в блоке логических операций может также выполняться операция сравнения.

4.4. Многофункциональные АЛУ

Операционные блоки для выполнения отдельных групп операций могут быть объединены в единую схему многофункционального АЛУ (рис. 4.21).

В состав схемы входят регистры Рг1, Рг2, Рг3, Рг3[1], РгА, РгВ и РгСм разрядностью 4 байта, а также регистры РгС, РгD и РгСч1 разрядностью 1 байт. Регистр РгСч1 является выходным регист-

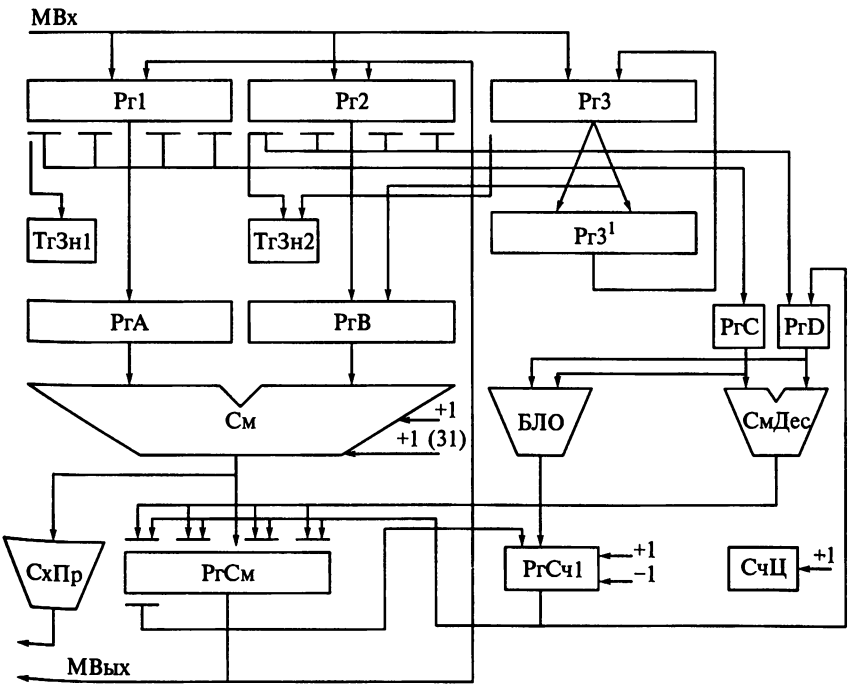


Рис. 4.21. Структура многофункционального АЛУ

ром блока логических операций и одновременно выполняет роль реверсивного счетчика. Загрузка операндов в АЛУ происходит по входной магистрали МВх, выдача результата — по выходной магистрали МВых. Использование регистров при выполнении операций в АЛУ показано в табл. 4.8.

Сложение-вычитание чисел с фиксированной точкой выполняется в основном сумматоре См. Слагаемые A и B заносятся в регистры Рг1 и Рг2 и далее на регистры РгА и РгВ. Операция выполняется в сумматоре См. Результат операции фиксируется в РгСм и может быть выдан на выходную магистраль данных.

Умножение и деление с ФТ выполняются с помощью одних и тех же узлов (см. рис. 4.15). Сдвиг множителя и частного осуществляется при передаче данных из Рг3 в Рг3[1]. При умножении имеется возможность прямой передачи данных из Рг3 в РгВ для передачи младших разрядов суммы ЧП или частного. Старшие разряды произведения фиксируются в регистре сумматора РгСм, младшие — в регистре Рг3. Частное формируется в регистре Рг3, откуда через регистр РгВ передается в РгСм. Результат операции выдается из РгСм на выходную магистраль.

При сложении чисел с ПТ слагаемые поступают на регистры Рг1 и Рг2. Далее знаки слагаемых передаются на триггеры знаков Тг3н1 и Тг3н2. Мантиссы слагаемых (M_A и M_B) передаются в регистры РгА и РгВ, а смещенные порядки (P_A и P_B) — в РгС и РгD. При выравнивании порядков в блоке логических операций производится сравнение порядков, затем мантисса числа с меньшим порядком сдвигается вправо (в регистре РгА или РгВ) с соответствующим увеличением порядка. Предельное число сдвигов мантиссы задается счетчиком циклов СчЦ. Порядок суммы фиксируется в регистре РгСч1. Сложение мантисс выполняется в сумматоре См. При необходимости производится нормализация результата и его округление. Для округления подается сигнал +1(31) в младший разряд сумматора.

Для умножения чисел с ПТ множимое Мм поступает на регистр Рг1, множитель Мт — на Рг3. Знаки сомножителей фиксируются в триггерах знака Тг3н1 и Тг3н2. Смещенные порядки множимого и множителя подаются на регистры РгА и РгВ соответственно и далее на сумматор для сложения порядков. Смещенный порядок произведения из регистра РгСм поступает на регистр РгСч1. При умножении мантисс мантисса множимого подается на регистр РгА, мантисса множителя — на регистр Рг3. Сумма ЧП мантисс накапливается в регистре РгВ. Число шагов при умножении задается при помощи счетчика циклов СчЦ. После перемножения мантисс при необходимости выполняется нормализация путем сдвига суммы ЧП в регистре РгВ с коррекцией смещенного порядка произведения в РгСч1. Старшие разряды мантиссы произведения передаются из РгВ в регистр сумматора РгСм,

Таблица 4.8

Размещение операндов в регистрах АЛУ

Тип операции	Рг1	Рг2	Рг3, Рг3[1]	РгА	РгВ	РгСм	РгС	РгD	РгСч1
Сложение чисел с ФТ	*A*	*B*	—	*A*	*B*	*A* + *B*	—	—	—
Умножение чисел с ФТ	Мм	—	Мт, младшие разряды произведения	Мм	Сумма ЧП	Старшие разряды произведения	—	—	—
Деление чисел с ФТ	Дт	—	Дм, Дм/Дт	Дт	Остаток	Остаток	—	—	—
Сложение чисел с ПТ	*A*	*B*	$М_A$	$М_B$	—	—	P_A	P	P_{A+B}
Умножение чисел с ПТ	Мм	—	Мт, младшие разряды произведения	$P_{Мм}$, $М_{Мм}$	$P_{Мт}$, ΣЧП	Старшие разряды $М_{Мм} \times М_{Мт}$	—	—	$P_{Мм \times Мт}$
Деление чисел с ПТ	Дт	—	Дм, $М_{Дм}/М_{Дт}$	$P_{Дт}$, $М_{Дт}$	$P_{Дм}$, остаток	Остаток	—	—	$P_{Дм/Дт}$
Операции десятичной арифметики	*A*	*B*	—	—	—	*A* * *B*	Байт *A*	Байт *B*	—
Логические операции	*A*	*B*	—	—	—	*A* * *B*	Байт *A*	Байт *B*	Байт *A* * *B*

в старший байт которого из РгСч1 заносится смещенный порядок произведения. Знак произведения формируется схемой формирования знака, не показанной на рис. 4.21, и заносится в знаковый разряд РгСм. Младшие разряды произведения остаются в регистре Рг3. Результат умножения выдается из РгСм на выходную магистраль Мвых.

При делении чисел с ПТ делимое заносится на Рг1, делитель — на Рг3. Далее знаки чисел передаются на триггеры знаков ТгЗн1 и ТгЗн2, цифровые разряды делимого заносятся на регистр РгВ, цифровые разряды делителя — на регистр РгА. Разность порядков определяется в сумматоре См и передается в регистр РгСч1 как порядок частного.

Мантисса частного определяется путем деления мантисс делимого и делителя. Мантисса делимого подается на сумматор из регистра РгВ, где в дальнейшем хранятся остатки, а мантисса делителя — из регистра РгА.

Разряды частного на каждом шаге заносятся в Рг3. Число шагов задается при помощи счетчика циклов СчЦ. После формирования мантиссы частного при необходимости выполняется ее нормализация. Для этого мантисса частного передается из Рг3 в РгВ, где выполняется ее сдвиг при одновременной коррекции смещенного порядка частного в регистре РгСч1. После нормализации формируется знак частного, который вместе с порядком частного передается в РгСм, где находится нормализованная мантисса частного. Из регистра сумматора частное выдается на выходную магистраль.

Операции десятичной арифметики выполняются при помощи десятичного сумматора СмДес разрядностью 1 байт. Байты операндов A и B из регистров Рг1 и Рг2 поочередно поступают на регистры РгС и РгD, переносы из младших байтов подаются на вход сумматора +1. Результат выполнения операции $A*B$ побайтно записывается на регистр сумматора РгСм.

Логические операции выполняются в блоке логических операций БЛО отдельно над каждым байтом операндов A и B. Байты операндов из регистров Рг1 и Рг2 заносятся на регистры РгС и РгD. Байты результата $A*B$ из регистра РгСч1 записываются в регистр сумматора РгСм.

При выполнении всех операций схема формирования признаков СхПр определяет значения признаков результата (флагов) и передает их в центральное устройство управления. Такими признаками могут быть наличие переполнения, равенство результата нулю, знак результата, наличие переноса (из байта или слова), четность числа единиц в двоичном коде результата и др. Признаки результата используются при выполнении операций условного перехода. Реакция компьютера на значения признаков зависит от ситуации и определяется ОС.

Контрольные вопросы

1. Какие операции могут выполняться в АЛУ?
2. Поясните особенности многофункциональных и блочных АЛУ.
3. Каковы особенности АЛУ магистрального типа и АЛУ с «жесткой» структурой?
4. Для чего используются признаки результата?
5. Как обнаруживается переполнение разрядной сетки при сложении чисел?
6. Какие возможные методы умножения чисел вы знаете?
7. Почему при умножении чисел в ДК необходима коррекция результата?
8. В чем заключается сущность логических методов ускорения умножения?
9. Каковы особенности деления без восстановления остатка?
10. В чем состоит общая идея логических методов ускорения деления?
11. Почему при сложении чисел с ПТ необходимо выравнивать порядки?
12. Как выполняется округление при сложении чисел с ПТ?
13. Почему возникает погрешность при выполнении операций с ПТ?
14. Почему при умножении и делении можно использовать одни и те же узлы АЛУ?
15. Для чего применяется счетчик циклов при умножении и делении?
16. В чем заключается особенность логических операций?

ГЛАВА 5

АРХИТЕКТУРА СОВРЕМЕННЫХ ПРОЦЕССОРОВ

5.1. Назначение и структура процессора

Современные процессоры строятся из одной или нескольких интегральных схем. Как правило, в персональном компьютере ЦП представляет собой одну микросхему, в которой используются все возможности современной полупроводниковой технологии. Такую микросхему принято называть *микропроцессором*. В больших компьютерах процессоры могут быть выполнены из нескольких интегральных схем.

В компьютерах используются процессоры нескольких типов. Наиболее распространены центральные, специализированные (например, процессоры для цифровой обработки сигналов или обработки графической информации), процессоры ввода-вывода, передачи данных и т. п.

Центральный процессор служит для обработки данных и управления всей компьютерной системой в целом. Он позволяет обрабатывать данные с фиксированной и плавающей точками, поля переменной длины, а иногда и десятичные данные, назначать приоритеты доступа и управлять различными видами памяти.

Процессор представляет собой совокупность АЛУ и устройства управления. В состав ЦП входят арифметическое устройство, счетчик команд, регистр команд, регистры данных, блок микропрограммного или аппаратного управления, регистры общего назначения и ряд узлов, предназначенных для связи с оперативной памятью и другими устройствами компьютера, а также для ускорения выполнения операций. Помимо этого в состав ЦП входят часы астрономического времени, таймер и ряд других «системных» средств.

В последнее время в одном кристалле с процессором начали размещать и кэш-память. Развитие технологии в конечном итоге приведет к тому, что в одной интегральной схеме будут интегрированы функции самых разных устройств, например в состав схемы процессора будет введена память. Для понимания принципов работы компьютера нужно разделять понятия «исполнение» и «выполняемые функции», поэтому процессором по-прежнему будем называть совокупность АЛУ и устройства управления.

5.2. Система команд. Форматы команд и способы адресации

Для получения результата вычислений компьютер выполняет *машинную программу*, т.е. последовательность команд, в виде которой записан алгоритм обработки. *Команда* компьютера представляет собой двоичный код, определяющий выполняемую операцию и необходимые для этого данные, или *операнды*. Большинство команд содержит *адрес*, т.е. номер регистра или ячейки памяти, где сохраняется результат выполнения операции. Для выполнения программы нужно также знать адрес следующей исполняемой команды.

Обычно различают команды:

арифметических операций над числами с ФТ;

логических операций;

арифметических операций над числами с ПТ;

операций ввода-вывода;

управления (например, управления циклами, условные и безусловные переходы) и т.д.

Каждый тип компьютера обладает собственной *системой команд*, т.е. в нем существует аппаратура или память микропрограмм, призванная вырабатывать управляющие сигналы с целью реализации командных операций. Для исполнения конкретной программы необходим компьютер, способный выполнять команды, составляющие эту программу. При разработке новых компьютеров стремятся сохранить их преемственность; для этого компьютеры изготавливают программно совместимыми, т.е. имеющими прежде всего одинаковые системы команд. Однако во многих случаях систему команд «расширяют», т.е. добавляют дополнительные операции, сохраняя при этом формат команд. Это значит, что новая машина может выполнять все программы, составленные для прежних компьютеров, но программы, в которых используются дополнительные команды, не могут выполняться компьютерами старых моделей. Такую совместимость называют *обратной*.

Команда, или *инструкция* (под инструкцией часто понимают конструкцию языка более высокого уровня, но в переводной литературе этот термин используют в качестве синонима термину «команда»), представляет собой слово, содержащее код выполняемой операции и адреса операндов. Код команды включает в себя несколько полей (поле — последовательность бит, содержащая определенный тип информации). Команда состоит из операционной и адресной частей. В *операционной* части размещается код операции, а в *адресной* — адреса операндов, т.е. информация о местонахождении обрабатываемых данных и получаемого результата.

Формат команды — это структура полей ее кода с указанием номеров разрядов, определяющих границы полей. Как правило,

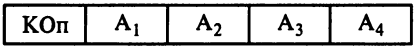


Рис. 5.1. Структура четырехадресной команды

в универсальных машинах код операции в команде занимает 8 разрядов, а число различных операций составляет не более 256. Остальные разряды отводятся под адреса операндов. Команды представляют собой слово размером 16, 32 или 48 разрядов.

В различных машинах число адресных полей в команде может составлять от одного до четырех. В четырехадресной команде (рис. 5.1) первое поле кода операции (КОп) предназначено для кодирования выполняемой операции. Это поле «расшифровывается» логическими схемами или микропрограммами, и формируются управляющие сигналы для выполнения соответствующих этой операции действий. Затем располагаются четыре адресных поля: A_1 — содержит адрес первого операнда, A_2 — адрес второго операнда, A_3 — адрес ячейки памяти, отведенной для записи результата операции, A_4 — адрес ячейки, где находится следующая команда. Но такая четырехадресная команда занимает слишком много места в памяти компьютера, поэтому они в настоящее время не находят применения.

Сегодня наибольшее распространение имеют одно-, двух- и трехадресные команды (рис. 5.2). Трехадресные команды характерны для компьютеров с сокращенным набором команд. Первый и второй адреса такой команды указывают месторасположение операндов, в третий адрес (ячейку памяти) заносится результат операции. Для определения адреса следующей выполняемой команды служит счетчик команд (IP), к содержимому которого после выполнения любой команды добавляется ее длина в байтах. Для перехода к выполнению команды, которая занимает не следующую по порядку ячейку памяти, в машине предусматривают специальные команды переходов. Трехадресные команды используются в так называемых RISC-компьютерах (машинах с сокращенным набором команд); в них операнды размещают в регистрах общего назначения, число которых может достигать 256. Загрузка этих регистров из памяти осуществляется специальной схемой, называемой *контроллером* или процессором загрузки.

Наиболее распространены в настоящее время двухадресные компьютеры. Это машины, команды в которых содержат не более двух адресов. Оба операнда находятся в регистрах, а результат вы-

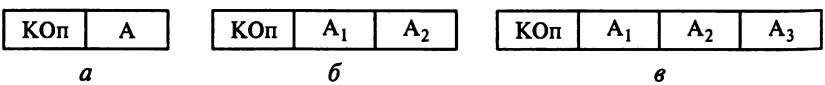


Рис. 5.2. Типовые форматы команд:
а — одноадресной; *б* — двухадресной; *в* — трехадресной

полнения операции также записывается в регистр. Такую команду принято называть командой RR-типа (регистр-регистр). Если один операнд находится в регистре, а второй в ячейке памяти, адрес которой индексируется, то такая команда относится к RX-типу. Команда, второй операнд которой находится в ячейке памяти без индексации, а первый в регистре, носит название RS-типа.

В команде может находиться не адрес операнда, а сам операнд (этот операнд называют непосредственным адресом, и он представляет константу), такую команду относят к SI-типу. Наконец, оба операнда могут находиться в памяти, для их вызова используют команду SS-типа.

В современном персональном компьютере IBM PC команды также двухадресные, но первый операнд всегда находится в одном из восьми регистров, а второй может находиться в регистре, памяти или непосредственно в самой команде. Помимо кода операции и адресов операндов команда этого компьютера содержит бит, указывающий направление передачи результата (*d*), бит ширины операнда (*w*), а также поле указания режима (*md*) и поле регистр/память (*r/m*).

В одноадресной машине команда содержит только один адрес, а поскольку обычно в арифметической операции используется два операнда, то второй уже находится в одном из регистров процессора. Результат операции всегда сохраняется в выходном регистре процессора, который называют *аккумулятором*. Для сложения двух чисел нужно выполнить три одноадресных команды: поместить первый операнд из памяти в регистр процессора, передать второй операнд из памяти и сложить его с первым, а затем сохранить результат в памяти. Однако на практике при выполнении программы никогда не приходится использовать все три команды. В регистре процессора сохраняется один из операндов, полученный при выполнении предыдущей операции (т. е. первая команда не нужна), а результат сложения будет использован следующей командой (т. е. его не нужно сохранять в памяти).

Приведенные на рис. 5.2 форматы команд весьма схематичны. В действительности в адресных полях находятся не сами адреса, а информация, позволяющая их определить. Способ определения исполнительного адреса $A_и$ (т. е. адреса ячейки памяти, где находится операнд или команда) по адресу в коде команды $A_к$ называется *способом адресации*.

Для указания способа адресации в некоторых системах команд используют специальное поле; это позволяет выполнять одну и ту же команду с любым предусмотренным способом адресации, но значительно увеличивает длину команды.

Обычно в команде имеются поля, указывающие адрес операнда $A_к$. В этом случае адресацию называют *явной*. Однако иногда адресное поле в команде отсутствует, а информацию об адресе опе-

ранда несет сам код операции. Такую адресацию называют *неявной*. Ее используют при работе над содержимым аккумулятора или флагами. *Флагами* называют признаки результата, которые при выполнении каждой операции заносятся в специальный регистр. Примерами могут служить флаги переполнения, четности и т. п. Общее число флагов зависит от конкретного процессора. Однако наиболее часто применяется явная адресация.

Время обращения в память сказывается на общей производительности компьютера, поэтому, чем меньше таких обращений, тем лучше с позиций быстродействия. Если нужно произвести действие с постоянным операндом, значение которого известно в момент трансляции программы, то нет необходимости хранить его в отдельной ячейке памяти; его можно поместить в адресном поле самой команды. Такой способ задания операнда принято называть *непосредственной* адресацией. Она применяется для задания констант, длина которых меньше адресного поля команды.

Операнды могут находиться и в регистровой, и в оперативной памяти; время обращения зависит от их местоположения. Если операнд находится в регистровой памяти, то время обращения к нему невелико. Именно для сокращения времени на обращение к операндам и используют *регистровую* адресацию. Регистровая адресация в машине IBM PC реализуется при $md = 1$. Адрес одного из восьми регистров, в котором размещается операнд, определяется полем *r/m*. Операнд может занимать весь регистр или только его половину в зависимости от значения *w*.

Наиболее часто используется *прямая* адресация. Операнд находится в ОП, а в команде указывается его исполнительный адрес (адресный код команды $A_к$ совпадает с исполнительным адресом $A_и$). В компьютерах IBM PC в этом поле команды находится не полный адрес, а только так называемое смещение, т. е. 16-битная часть исполнительного адреса. В одном из адресных регистров находится адрес сегмента; исполнительный адрес определяется сложением номера (адреса) сегмента и смещения. Этим удалось избежать самого крупного недостатка прямой адресации, а именно, необходимости слишком длинного адресного поля при большой емкости памяти.

При *косвенной* адресации в адресном поле команды указывается адрес регистра или ячейки памяти, где находится адрес операнда; этот адрес называют *указателем*. В случае когда указатель располагается в регистре, адресацию называют *косвенно регистровой* (рис. 5.3).

Адрес указателя в команде компьютера остается неизменным, но, меняя сами указатели, можно осуществлять переадресацию данных, что упрощает обработку массивов и списковых структур.

Часто адрес ячейки памяти образуется из нескольких составляющих частей, например базового адреса (или базы *B*), индекса

(X) и смещения (D). В этих случаях различают базовую, индексную и базово-индексную адресации. Если адрес, указанный в команде, состоит из двух частей — A_B и A_D, то полный адрес ячейки памяти определяется суммированием содержимого регистра A_B, где хранится база, и смещения A_D (рис. 5.4).

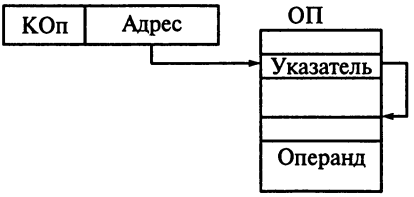


Рис. 5.3. Косвенная адресация

Базовая адресация широко используется при создании программ, состоящих из перемещаемых модулей. Базовый (начальный) адрес загружается в регистр, а остальные адреса данного модуля определяются сложением базы и смещения. Это значит, что любая программа может работать с данными, располагаемыми в любой области памяти; для этого нужно изменить только базу.

Аналогично выполняется и *индексная* адресация. Однако после выполнения обращения к ячейке памяти с адресом $A = A_X + A_D$ производится увеличение индекса X, записанного в индексном регистре, на единицу. Так «автоматически» определяется адрес следующего подлежащего обработке элемента в массиве. Использование индексной адресации упрощает создание циклических программ.

Часто используют *комбинированную* базово-индексную адресацию. В этом случае адрес ячейки памяти определяется суммой содержимого регистров базы и индекса, а также смещения, содержащегося в команде (рис. 5.5). Иногда используют не сложение, а

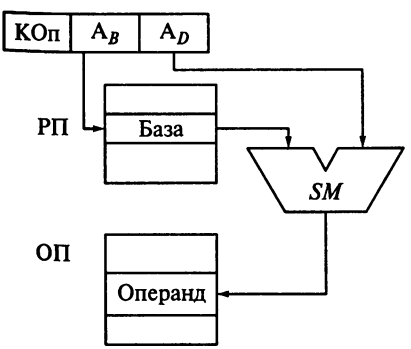


Рис. 5.4. Формирование адреса памяти при базовой адресации

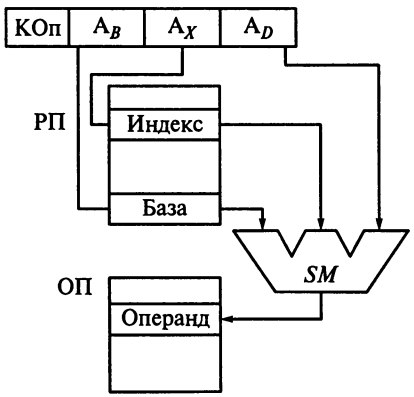


Рис. 5.5. Формирование адреса памяти при базово-индексной адресации

«сцепление» кодов базы, индекса и смещения. Это значит, что исполнительный адрес $A_И = A_B, A_X, A_D$.

Для определения адреса следующей команды в IBM PC может быть использована *относительная* адресация. В этом случае адрес очередной команды представляет собой сумму текущего значения счетчика команд (IP) и смещения из предыдущей команды.

Все виды адресации в компьютере IBM PC определяют место расположения и способ нахождения адреса второго операнда; первый операнд всегда находится в одном из регистров. Для компьютеров IBM PC многочисленные способы адресации отчасти достались «по наследству» от ранних моделей микропроцессоров, когда разрядность была ограничена.

Выбору реализуемой компьютером системы команд должно уделяться самое серьезное внимание. Каждая аппаратно реализуемая операция, входящая в систему команд, выполняется быстрее, чем аналогичная операция, не входящая в систему команд и, следовательно, реализуемая в виде подпрограммы. На первый взгляд из этого следует, что увеличение аппаратно реализуемых операций, т.е. расширение системы команд, может привести к повышению быстродействия машины. Этот вывод положен в основу концепции архитектуры компьютера с расширенным набором команд — CISC-архитектур. Однако расширение системы команд может приводить и к обратным результатам. Именно это обстоятельство послужило толчком для разработки машин с сокращенным набором команд — RISC-архитектур.

5.3. Система прерываний и приостановок, состояние процессора

Во время выполнения какой-либо программы компьютером могут возникнуть события, требующие его немедленной реакции. Необходимость незамедлительного решения другой задачи, переполнение разрядной сетки, программный или аппаратный сбой, окончание предусмотренного интервала времени и т.д. — все это события, требующие переключения компьютера на другую программу. Переход к другой программе осуществляется посредством *системы прерываний.*

Прерывание программы — это процесс переключения процессора с одной программы на другую по внешнему сигналу с сохранением информации для последующего возобновления прерванной программы. При возникновении события, приводящего к прерыванию, формируется сигнал, называемый *запросом прерывания.* Существует несколько источников запросов прерывания: это схемы контроля процессора, система питания, память, периферийные устройства и т.д. При наличии нескольких источников запро-

сов прерывания устанавливается определенный порядок их обслуживания путем назначения приоритетов. Запросы прерываний направляются на различные разряды специального регистра (регистра запросов прерываний), опрос которого производится в строго определенной последовательности при завершении очередной команды. Номер разряда этого регистра не только определяет приоритет запроса прерываний, но и позволяет найти соответствующую данному запросу программу обслуживания прерывания. Поступивший запрос может прервать только менее приоритетную программу.

Существует также векторная система прерываний. В ней информация о месте возникновения запроса передается от источника прерываний в виде адреса ячейки памяти, содержимое которой определяет конкретную программу обслуживания. Помимо адреса перехода к программе обслуживания эта ячейка хранит дополнительную управляющую информацию. Содержимое этой ячейки (или нескольких последовательных ячеек) принято называть *вектором прерываний.*

Еще один очень важный процесс — *приостановки*, при которых средства управления, работающие автономно от процессора, задерживают его работу на время цикла памяти, когда память занята приемом или выдачей информации для другого устройства. Во время приостановок, называемых также занятием цикла памяти, процессор никаких действий не выполняет, его состояние не меняется, но выполнение очередной команды задерживается до освобождения памяти. Возможности ограничены непосредственной передачей данных между ОП и процессором, когда память или шина используются несколькими устройствами.

Работой компьютера управляют команды программы, но конкретные выполняемые действия зависят от его текущего состояния. При работе машины в специальном регистре процессора постоянно находится *слово состояния программы*, которое и характеризует его состояние. Это слово содержит информацию, необходимую для возобновления программы при прерываниях: указания о разрешенных прерываниях, адрес текущей выполняемой команды, различные признаки, ключи защиты и маски.

Характеристики системы прерываний. Систему прерываний характеризуют: общим числом входов от источников (числом запросов прерываний); числом уровней прерывания, по которым сгруппированы источники, вызывающие одну и ту же прерывающую программу; глубиной прерывания, т.е. максимальным числом программ, которые могут быть последовательно прерваны друг другом; системой приоритетов и организацией переходов к следующей программе.

Одной из основных характеристик системы прерываний служит *время реакции* $\tau_{ож}$, т.е. время ожидания, прошедшее между

запросом прерывания и началом переключения программ. Это время зависит не только от характеристик системы прерывания, но и от числа ожидающих обслуживания программ со старшими приоритетами. По этой причине время реакции определяют для запроса с наибольшим приоритетом, который будет обрабатываться первым. Обычно наивысший приоритет назначают запросам прерываний, поступающим от аппаратуры контроля, чтобы обнаруженная ошибка не оказала влияния на последующую работу компьютера.

В персональных компьютерах, как правило, процесс прерывания происходит по окончании текущей команды (рис. 5.6), и время реакции определяется длительностью этой команды. На этом рисунке $\tau_з$ — время запоминания параметров текущей команды, $\tau_{п.п}$ — время выполнения прерывающей программы и $\tau_в$ — длительность восстановления параметров текущей программы. Если компьютер предназначен для систем реального времени, то это время может оказаться недопустимо большим, поэтому прерывание выполняется после каждой микрокоманды. Однако при этом количество запоминаемой и восстанавливаемой информации значительно возрастает.

Для уменьшения времени на переключение программ состояние прерванной программы запоминают в специальной стековой памяти. Очевидно, что это время зависит от количества запоминаемой информации.

Затраты времени на запоминание параметров текущей программы (для освобождения ресурсов процессора) $\tau_з$ и восстановление состояния процессора после выполнения прерывающей программы $\tau_в$ определяют длительность переключения программ при прерывании. Эта длительность обычно составляет несколько машинных тактов.

Одной из основных характеристик системы прерываний является *глубина прерываний* — максимальное число программ, способных прерывать друг друга. В простейших компьютерах глубина

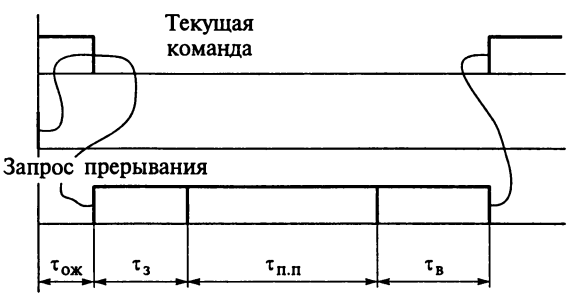


Рис. 5.6. Временная диаграмма процесса прерываний

обычно равна 1, т.е. после начала выполнения прерывающей программы запросы на прерывание не обслуживаются. В большинстве компьютеров глубина прерываний больше 1 и более приоритетные запросы обслуживаются в первую очередь. Так, если текущая программа прервана программой ввода-вывода от клавиатуры, а в процессе ее выполнения поступил запрос от накопителя на дисках, то она также будет прервана. После этого завершится программа ввода-вывода от накопителя, затем программа ввода-вывода от клавиатуры, и только после этого первая программа.

Если запрос на прерывание от какого-либо источника не будет обслужен до прихода очередного запроса от того же источника, то он теряется; возникает насыщение системы прерываний.

Приоритет прерываний. Во многих компьютерах, особенно предназначенных для управления производственными процессами, число различных источников прерывания может достигать нескольких сотен, а их обслуживание вызывает сложности.

Для упрощения обработки прерываний эти источники подразделяют на отдельные уровни. Всем запросам, обрабатываемым одной программой обработки прерываний, присвоен один уровень. Запросы (сигналы) поступают на определенные разряды регистра, устанавливая их в «1». Выходы нескольких разрядов этого регистра объединены схемой ИЛИ, сигнал с которой вызывает нужную программу обработки.

Поскольку существует несколько источников прерываний, например схемы контроля ЦП, системы питания, памяти, внешние события и т.п., сигналы от которых могут поступить одновременно, то нужно установить очередность обработки этих прерываний. Порядок обслуживания прерываний устанавливается путем назначения приоритетов. Запросы прерываний в зависимости от назначенного приоритета направляются на различные разряды регистра прерываний процессора, опрос которого производится в строго определенной последовательности. Так, запрос самого высокого уровня поступает на первый «опрашиваемый» разряд этого регистра и, следовательно, всегда обрабатывается первым. Самым высоким приоритетом обладают прерывания от схем контроля. Поступивший запрос прерывания может прервать только менее приоритетную программу.

Время задержки в обслуживании прерывания определяется не только реакцией системы, но и числом ожидающих запросов более высокого приоритета.

Организация перехода к прерывающей программе. Векторная система прерываний широко применяется в персональных компьютерах. После обнаружения сигнала прерывания при выполнении каждой команды процессора производится проверка — разрешено ли оно. Прерывание может быть замаскировано. Маскирование прерываний позволяет защитить критические секции теку-

щей программы, т.е. участки программы, на выполнение которых не должно оказывать влияние изменение содержимого памяти, вызываемое процессором ввода. Если обнаружено разрешенное прерывание, то начинается его обслуживание.

5.4. Режимы работы процессора

При наличии единственного процессора компьютер в каждый момент времени может выполнять лишь одну команду программы. Однако в память компьютера может быть введено несколько программ. При *однопрограммном* режиме работы переход к выполнению следующей программы осуществляется только после полного завершения предыдущей. Компьютер, способный работать исключительно в однопрограммном режиме, обладает наиболее простой структурой. Именно по этой причине все первые персональные компьютеры были однопрограммными.

Однако выполнение программы требует не только ресурсов процессора. Нужно, чтобы в оперативной памяти присутствовали как данные, так и программа, а обработка не задерживалась из-за отсутствия в ОП необходимой информации или занятости областей памяти результатами предыдущей обработки. Ввод и вывод информации осуществляются соответствующими устройствами, например клавиатурой, принтером и т.п. При вводе и выводе ресурсы процессора (например, арифметическое устройство) практически не используются и, следовательно, могут быть доступными для другой программы.

Для этого в память компьютера должно быть введено несколько программ. Они могут находиться в состоянии *обработки* (активное состояние, при котором программа обрабатывается в процессоре), *готовности к обработке* или *ожидания* некоторого события (завершения операции ввода-вывода или освобождения ресурса).

Когда для текущей программы не хватает данных, процессор переключается на выполнение другой готовой для обработки программы, а текущая программа откладывается. Такой режим называют *мультипрограммным*; он предполагает наличие нескольких независимых программ (или фрагментов программы), принятых на обслуживание. При этом фрагменты программы считают независимыми, если каждый их них может быть выполнен без результатов обработки других. При мультипрограммном режиме обслуживание, т.е. обработка, ввод или вывод любой программы, может быть начато независимо от завершения других фрагментов. Все программы (или запросы на выполнение) находятся в очередях к соответствующим устройствам: процессору, внешней памяти, устройствам ввода или вывода.

Переключение программ из состояния готовности в состояние обработки требует дополнительного времени для процессора на выполнение управляющей программы. Кроме того, необходимы дополнительные средства для сохранения промежуточных результатов, ведения очередей и т. п. Но за счет мультипрограммирования удается значительно повысить загрузку процессора, а следовательно, и пропускную способность компьютера. Скорость работы процессора остается прежней, но за счет его равномерной и более плотной загрузки повышается пропускная способность компьютера.

На рис. 5.7 показан пример ускорения выполнения трех программ A, B и C за счет параллельного выполнения операций обработки и ввода-вывода. Для сравнения на рис. 5.7, *а* показано выполнение программ в последовательном однопрограммном режиме. Все программы разбиты на участки ввода (A_1, A_3, B_1, C_2), обработки (A_2, A_4, B_2, C_1, C_3) и вывода (A_5, B_3, C_4). Для сравнения очередность выполнения участков принята одинаковой. Дополнительные затраты времени $\tau_д$ (см. рис. 5.7, *б*), необходимые на переключение программ, условно показаны в виде одного интервала в конце выполнения программ (а не в виде отдельных интервалов перед каждым участком программ), общая длительность — T.

При мультипрограммном режиме возникают дополнительные трудности в случае организации вывода на коллективно используемое периферийное устройство, например на единственный принтер, который печатает результаты работы всех программ. Так, прежде чем начать печатать результаты решения очередной зада-

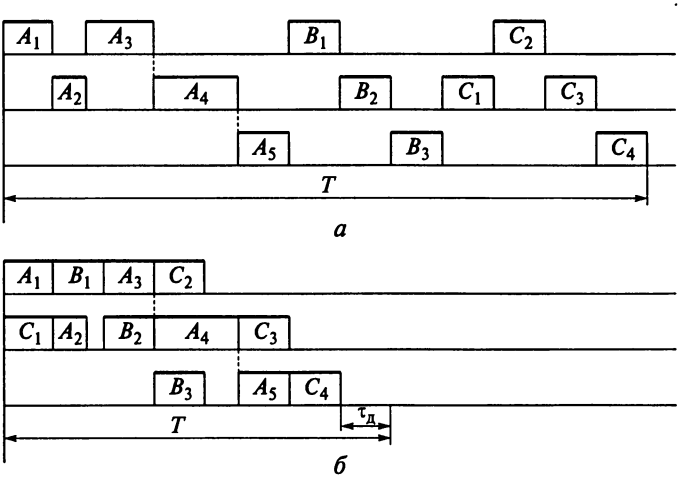


Рис. 5.7. Пример выполнения трех программ:
а — в однопрограммном режиме; *б* — в мультипрограммном

чи, необходимо полностью закончить печать результатов предыдущей. По этой причине организуют так называемый системный вывод, т.е. результаты обработки заносят в буферные области памяти, отведенные для каждой задачи, а вывод результатов на коллективно используемый принтер выполняют из этих областей только после его освобождения.

Таким образом, выполняемые в мультипрограммном режиме программы конкурируют за получение времени процессора, доступа к памяти, устройствам ввода и вывода. Распределение ресурсов системы между выполняемыми программами осуществляется управляющими программами ОС, которые переключают программы и предоставляют им необходимые ресурсы.

Известно несколько режимов обработки программ. В системах общего назначения наиболее часто используют *пакетный* режим. Пакет заданий (совокупность независимых друг от друга программ) вводят в память компьютера. Каждое задание снабжается описанием, содержащим приоритет задания, необходимую емкость памяти и требуемые устройства ввода-вывода. Задания, имеющие одинаковый приоритет, попадают в одну очередь. Задача из очереди с наибольшим приоритетом, находящаяся в состоянии готовности, переходит в активное состояние и начинает обрабатываться. Этот процесс будет прерван, если необходимые для решения задачи ресурсы недоступны, а также при появлении готовых к выполнению более приоритетных задач. Однако пакетный режим решения задач обладает и рядом недостатков. Время ожидания результатов решения задачи заранее не определено; помимо ресурсов компьютера оно зависит от числа входящих в пакет программ и их приоритетов, приоритета самой задачи и других факторов. Невозможно оперативно вносить изменения в программу; любое изменение связано с необходимостью повторного ввода нового пакета. Все это привело к появлению компьютеров, работающих в режиме *разделения времени*. Компьютер, работающий в этом режиме, должен предоставлять каждому пользователю терминал, посредством которого пользователь может передать запрос на обработку своей задачи и получить результаты ее решения. Процессор выделяет каждому запросу для его обработки некоторый интервал времени. Если длительность этого интервала недостаточна для завершения обработки запроса, то такая программа прерывается (независимо от степени готовности решения) и опять направляется в очередь, а ресурсы процессора предоставляются следующей программе. Длительность этих интервалов обслуживания выбирается так, чтобы сделать приемлемыми время ожидания ответа на запросы и затраты времени на переключение программ.

Помимо выбора длительности интервала необходимо распределить их между отдельными программами. Такое распределение

устанавливается в соответствии с принятой дисциплиной обслуживания. В простейшем случае все вновь поступающие запросы пользователей ставятся в конец общей очереди, а для обслуживания выбирается программа из ее начала. Если за выделенный интервал времени программа успевает завершиться, то результат выдается пользователю. Если же по истечении этого интервала выполнение программы не завершено, она направляется в конец очереди, а ресурсы компьютера предоставляются следующей программе из очереди.

Для уменьшения потерь времени на переключение программ и первоочередное предоставление ресурсов более коротким программам используют дисциплину обслуживания с несколькими очередями, для которых выделяемые интервалы времени переменны. Поступающая на обработку программа ставится в конец очереди с наибольшим приоритетом. Если эта очередь пустая, то происходит выбор задачи из следующей очереди, а если программа за выделенный интервал времени не завершена, она переходит в конец следующей очереди. Задачи, стоящие в последней предусмотренной очереди, выполняются до конца без прерывания. Иногда приоритет делается зависящим от времени ожидания. Тогда программа может переходить из одной очереди в другую, а приоритет становится динамическим.

Компьютеры, предназначенные для систем цифровой обработки сигналов, работают в режиме *реального времени*. Существует множество задач фильтрации сигналов, спектрального анализа и синтеза, распознавания речи, управления производственными процессами и ряда других, решение которых должно быть получено не позже определенного момента времени. Данные для них поступают и выдаются по каналам связи в цифровом или аналоговом виде. Компьютер работает в реальном масштабе времени, если он формирует решение не позже заранее установленного срока. Быстродействие такого компьютера зависит от управляемого процесса; оно может быть и очень большим, и сравнительно невысоким.

Для поддержания режима реального времени в компьютере должен быть таймер, позволяющий измерять интервалы времени между различными событиями. Однако таймер устанавливается не только в компьютерах реального времени. Он необходим и для компьютеров, работающих в пакетном режиме и режиме разделения времени. В компьютерах, работающих в пакетном режиме, переключение программ производится не только тогда, когда программа не может выполняться из-за нехватки ресурсов, но и по истечении некоторого интервала времени. В режиме разделения времени необходимо измерять предоставляемый каждой программе интервал. Кроме того, обычно ведется учет времени, использованного каждой программой.

Наличие нескольких выполняемых программ приводит к необходимости организации защиты памяти, предотвращающей воздействие одной программы на другую. Каждая программа получает в свое распоряжение определенную область памяти, а вторжение в «чужую» область приводит к искажению информации, принадлежащей другой программе. Для предотвращения этого в компьютерах предусматривают средства защиты памяти.

Рассмотренные режимы в большей степени характерны для ЭВМ общего назначения или управляющих компьютеров. Но и в персональных компьютерах реализуются мультипрограммные режимы работы: компьютер может выполнять задачу пользователя, общаясь с ним в интерактивном режиме, и одновременно решать «фоновую» задачу, обычно требующую значительного времени для своего решения.

5.5. Компьютеры CISC и RISC

Существует два подхода к повышению производительности компьютера: создание машин с архитектурой CISC (Complex Instruction Set Computer — компьютер со сложным набором команд) и RISC (Reduced Instruction Set Computer — компьютер с сокращенным набором команд). Оба подхода обладают достоинствами и недостатками, поэтому в последних процессорах, например Pentium IV, принято компромиссное решение.

В компьютерах с архитектурой CISC увеличение производительности достигается за счет расширения числа аппаратно и микропрограммно реализуемых операций. Считалось, что расширение системы команд за счет включения в нее все более сложных операций представляет один из признанных архитектурных принципов повышения быстродействия процессора, поэтому все мощные компьютеры строились с архитектурой CISC. Для такой архитектуры характерно:

большое число машинных операций (именно поэтому архитектура и названа CISC), некоторые из которых для своего выполнения требуют нескольких тактов, так как для реализации этих операций широко используется принцип микропрограммирования;

разнообразие способов адресации, часто приводящих к необходимости нескольких обращений в память при выполнении одной команды;

преобладание двухадресных команд, причем в команде прямо указывается адрес памяти (или регистра), куда следует поместить результат;

наличие сравнительно небольшого числа регистров общего назначения, что вызвано ограничениями на длину команды;

широкое использование микрокомандного принципа управления.

Однако введение в систему команд таких команд, которые для своего выполнения требуют нескольких машинных тактов, имеют различную длину и используют разные способы адресации, сильно усложняет логику управления и весь процессор в целом. На пути повышения производительности машины за счет расширения системы ее команд возникает, таким образом, ряд трудностей.

Во-первых, реализовать сложную систему команд с помощью только аппаратных средств довольно трудно, поэтому для ее реализации почти всегда используют микропрограммные средства, т.е. дополнительную память с соответствующими каждой сложной операции микропрограммами. В этом случае каждая команда выполняется как бы за два этапа: первый этап связан с получением команды и операндов из памяти, а второй — с выполнением микропрограммы, хранящейся в памяти микропрограмм. Такое двухэтапное выполнение команды замедляет ее выполнение.

Во-вторых, микропрограммы «сложных» команд занимают значительную часть управляющей памяти.

В-третьих, аппаратная реализация редко встречающихся операций не приводит к заметному увеличению быстродействия процессора, но значительно усложняет его структуру и отдельных интегральных схем.

В-четвертых, усложнение интегральных схем означает увеличение числа вентилей и длины проводников, т.е. ведет к увеличению задержек сигналов, а следовательно, к снижению скорости выполнения операций.

Все это обусловило появление архитектуры с сокращенным набором команд — RISC-архитектуры, в которой команды извлекаются из памяти в виде равномерного потока, а процессор постоянно загружен. Для повышения быстродействия при сокращении набора команд служат следующие основные предпосылки:

подавляющее большинство команд (более 90 %) при решении определенной задачи образует компактное множество; команды, не входящие в это множество, при решении данной задачи встречаются чрезвычайно редко;

сравнительно небольшой набор команд можно эффективно реализовать аппаратными средствами так, что каждая операция реализуется за один такт;

для современного этапа развития технологии характерно заметное уменьшение различия в быстродействии оперативной и постоянной памяти микропрограмм, что делает многоуровневый принцип программирования нецелесообразным;

современный уровень развития технологии позволяет на одном кристалле микропроцессора размещать большое число реги-

стров, так называемый регистровый пул, а это делает возможным обращение к регистрам, а не к оперативной памяти.

Для RISC-процессоров характерно:

малое число операций, причем каждая из них выполняется аппаратными средствами строго за один такт;

постоянный формат команд, что ускоряет расшифровку команд и делает возможным быстро получать их из памяти (обычно число форматов команд и способов адресации не превышает четырех);

преобладание трехадресных команд;

неразрушающий принцип обращения к ОП, т.е. полученный результат выполненной операции не затирает значения операнда в памяти, а помещается в отдельный регистр;

разделение команд обработки (эти команды имеют формат RR) и обмена с памятью (формат RS), причем команды обмена с памятью выполняются независимым блоком;

наличие большого регистрового пула, т.е. число регистров общего назначения составляет не менее 32, а в большинстве современных RISC-компьютеров 256 (известны компьютеры и с большим числом регистров);

использование принципа «жесткой» логики для схем управления процессором.

Система команд RISC-процессора включает в себя небольшое (по сравнению с CISC) число операций и использует ограниченное число способов адресации; к этим командам относятся команды работы с памятью, копирования и некоторых арифметических операций. Поскольку число команд небольшое, то длина программы увеличивается и, как показали исследования, она становится примерно на 30 % длиннее аналогичной программы CISC-компьютера.

Команды обращения к памяти выполняются отдельным блоком и служат для загрузки регистров процессора, число которых может достигать нескольких десятков и даже сотен. Очень важно, что загрузка регистров из памяти, а также передача их содержимого в память осуществляется совершенно независимо от работы собственно процессора. Все регистры объединены в несколько *перекрывающихся окон* — регистры одного из окон загружаются из памяти, в то время как регистры из другого окна служат для работы с процессором. Перекрытие окон обеспечивает возможность непрерывного размещения команд и взаимодействия процессора со средствами загрузки. Большое число регистров дает неоспоримые преимущества, но усложняет схемы декодирования номера регистра, что приводит к некоторому увеличению времени обращения к ним.

Команды RISC-процессора обычно имеют фиксированный формат, что позволяет извлекать команду из памяти за одно обра-

щение. Однако фиксированный формат исключает прямую адресацию памяти. Дешифрация кода операции, осуществляемая исключительно аппаратными средствами, также выполняется за один такт. Такие процессоры используют трехадресные операционные команды типа RR и содержат кэш-памяти достаточно большого объема. Все это приводит к ускорению выполнения программ, несмотря на то что их длина (т. е. число команд) несколько увеличивается.

В RISC-процессорах реализуются средства повышения производительности, характерные для CISC-систем: суперскалярность (несколько команд выполняются за один машинный такт) и предсказание переходов.

Для RISC-процессоров характерно наличие средств поддержки мультипроцессности и компиляторов, позволяющих выполнять оптимизацию программ. Кроме того, на компилятор возлагаются дополнительные функции по контролю над вычислительным процессом, в том числе:

распределение регистров, уменьшающее число команд пересылки данных между регистрами и памятью;

защита памяти;

контроль переполнения;

предотвращение блокировок конвейеров и т. п.

Компилятор должен так формировать программу, чтобы признаки результата использовались не следующей по порядку командой, а через одну (или две).

В настоящее время большинство процессоров персональных компьютеров выполняют в виде одной или, реже, нескольких интегральных схем, которые принято делить на процессоры общего назначения и специализированные. К числу процессоров общего назначения относятся популярные процессоры Pentium фирмы Intel, большинство процессоров фирмы AMD, а также фирмы Apple. Из-за большой сложности таких процессоров число изготавливающих их фирм очень мало. Процессоры персональных компьютеров изготавливаются в виде БИС и требуют сложнейшего оборудования стоимостью в сотни миллионов долларов.

Для повышения производительности в микропроцессорах персональных компьютеров используют технические решения, подобные тем, что применяются в современных быстродействующих RISC-процессорах. Наиболее часто для повышения производительности процессоров в персональных компьютерах используют:

суперскалярную обработку данных, предусматривающую запуск и параллельное выполнение нескольких команд из одной программы;

увеличение разрядности шины для передачи данных и команд между процессором и памятью до 64 бит, а иногда и больше;

внутреннюю кэш-память команд и данных первого уровня достаточно большой емкости;

внешнюю кэш-память второго уровня, емкость которой составляет несколько сотен килобайт.

Кроме того, в большинстве этих микропроцессоров предусматривается:

изменение последовательности выполнения команд, т.е. выполнение команд в последовательности, отличной от той, в которой они находятся в программе. Это повышает производительность процессора, так как ему не нужно ждать результатов предыдущих операций для выполнения последующих;

переименование регистров, позволяющее «увеличить» размеры блока регистров, т.е. ослабить влияние самого главного недостатка архитектуры CISC.

Однако перед рассмотрением перечисленных механизмов повышения производительности нужно познакомиться с функциями и структурой устройства управления.

5.6. Устройства управления

Любое устройство компьютера состоит из двух блоков — операционного и управляющего. Функции управления ходом вычислительного процесса возлагаются на устройство, или блок, управления. Оно предназначено для получения информации о ходе вычислительного процесса и выработки сигналов для его продолжения. Элементы управления присутствуют во всех узлах и устройствах компьютера, поэтому конструктивно БУ выполняется в виде центрального блока, призванного формировать управляющие сигналы, и отдельных схем, распределенных по всему компьютеру и служащих для использования этих сигналов.

Входной информацией для центрального БУ служит *код операции* текущей команды, который определяет конкретную выполняемую операцию. Он поступает в БУ из регистра команд и служит для выработки сигналов управления, передаваемых на все узлы и устройства компьютера.

Помимо кода операции при работе БУ используются *флаги*, т.е. признаки, характеризующие результаты выполнения предшествующей операции. Они необходимы для организации выполнения команд условного перехода. Блок управления вырабатывает сигналы при поступлении на него тактовых импульсов. Кроме того, он получает сигналы от системной шины, например запросы прерывания, подтверждения приема информации и т.п.

Блок управления формирует управляющие сигналы, направляемые в различные устройства и блоки компьютера, прежде всего в процессор. Эти сигналы служат для управления перемещением

данных между регистрами, работой устройств, инициирования тех или иных функций. Сигналы управления направляются также в шину для передачи в блоки памяти и на устройства ввода-вывода.

Действия в операционном блоке в течение одного такта называются *микрооперациями*, а любая операция или команда, выполняемая за несколько тактов, описывается некоторой *микропрограммой*, определяющей последовательность действий при выполнении этой команды. Сигналы управления, формируемые БУ и поступающие на входы операционного устройства, вызывают в нем выполнение микрооперации. Поскольку блоки управления формируют сигналы для выполнения микроопераций, их часто называют *микропрограммными автоматами.*

В зависимости от способа формирования сигналов управления микрооперациями различают два типа микропрограммных автоматов: с «жесткой» и программируемой логикой.

Устройства управления с «жесткой» логикой. Код операции и номер такта в управляющем автомате с «жесткой» логикой (рис. 5.8) поступают на дешифраторы *DC* и затем на логические схемы формирования сигналов управления. На эти же схемы, представляющие собой наборы схем совпадения, поступают различные осведомительные сигналы, которые характеризуют результат выполнения предыдущей операции или микрооперации.

Сигналы для каждой микрооперации в автомате с «жесткой» логикой формируются своим собственным набором логических схем, который невозможно изменить, не меняя всю систему управления компьютером, что и обусловливает название автомата. В состав его схемы входят счетчик и дешифратор тактов, дешифратор кода операции и логические схемы формирования сигналов управления. Время срабатывания этих логических схем мало, поэтому такие аппаратно реализованные микропрограммные автоматы обеспечивают наибольшее быстродействие. Однако их сложность возрастает с расширением системы команд, так как для каждой операции в них должен быть предусмотрен собственный набор схем.

Эти особенности автоматов с «жесткой» логикой привели к тому, что они находят применение в RISC-процессорах, а в компьютерах с расширенной системой команд обычно используют иной подход — микропрограмму, хранимую в специальной памяти.

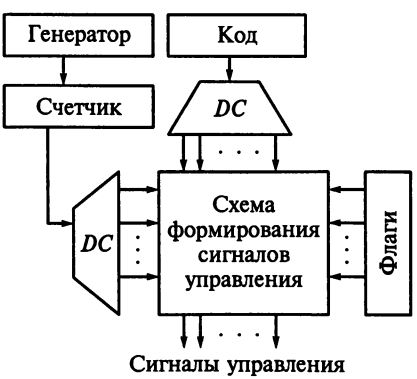


Рис. 5.8. Структура управляющего автомата с «жесткой» логикой

Устройства управления с хранимой в памяти логикой. Второй тип управляющего автомата — автомат с хранимой в памяти логикой. Управление каждой операцией, входящей в систему команд компьютера, осуществляется с помощью хранимых в памяти слов. Эти управляющие слова, или *микрокоманды*, содержат информацию о микрооперациях, выполняемых в течение одного машинного такта и осуществляющих элементарное преобразование над данными, и указание, где находится следующая микрокоманда. Такое управление получило название *микропрограммного.*

Одна машинная команда CISC-компьютера обычно выполняется в течение нескольких тактов, а управление осуществляется последовательностью микрокоманд, составляющих микропрограмму. Микропрограммы для каждой команды компьютера можно разместить в специальной *памяти микропрограмм.* Таким образом, процесс командного управления становится двухступенчатым: сначала нужно извлечь из основной памяти компьютера команду, а затем для ее выполнения извлечь из памяти микропрограмм последовательность микрокоманд. Для реализации такого двухступенчатого процесса требуется быстрая постоянная память довольно большого объема, которая появилась в середине 1960-х гг. Начиная с этого времени устройства управления с хранимой в памяти логикой стали применяться в большинстве процессоров общего назначения малой и средней производительности, а также в управляющих процессорах, некоторых микропроцессорах и различных контроллерах периферийных устройств.

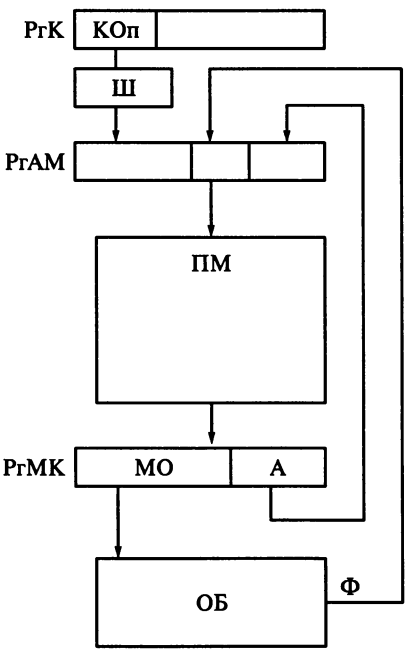


Рис. 5.9. Структура устройства управления с хранимой в памяти логикой

Основное место в типичной структуре устройства управления с хранимой в памяти логикой (рис. 5.9) занимает память микропрограмм (ПМ), в которой находятся все выполняемые микрокоманды. Она является постоянной памятью; адрес очередной микрокоманды (РгАМ) определяется кодом текущей операции (КОп из регистра команд преобразуется шифратором Ш), флагами Ф, полученными при выполнении предыдущей

операции (из регистра флагов), и адресной частью предыдущей микрокоманды (А).

Очередная микрокоманда считывается из памяти и заносится в регистр (РгМК), откуда сигналы управления микрооперациями (МО) поступают в операционный блок (ОБ). Все действия в микропрограммном блоке синхронизируются управляющими синхросигналами, не показанными на рисунке. Управляющие сигналы в микрокомандах, хранящихся в ПМ, могут быть представлены различными способами: горизонтальным, вертикальным и смешанным. Способ кодирования микроопераций определяет сложность и параметры быстродействия блока управления.

При *горизонтальном* способе каждый управляющий сигнал кодируется одним разрядом в микрооперационной части микрокоманды. Этим достигается максимальный параллелизм формирования сигналов управления, а следовательно, и выполняемых действий. Однако при горизонтальном способе длина микрооперационной части может достигать сотен бит, причем большинство из них содержат нули.

При *вертикальном* способе каждой микрооперации соответствует некоторый код, который и хранится в микрооперационной части микрокоманды. Тогда микрокоманда имеет минимальную длину. Но теперь для формирования сигнала управления нужен дешифратор микрокоманд и, кроме того, в каждой микрокоманде можно «закодировать» лишь один сигнал управления. Таким образом, увеличиваются длина микропрограммы и время ее выполнения.

При *смешанном* кодировании все микрооперации разбиваются на группы. В каждую группу можно включить сигналы управления, которые никогда не встречаются вместе (это так называемый горизонтально-вертикальный способ). При этом способе используемые в одном такте сигналы формируются в разных группах. Сигналы управления внутри группы кодируются вертикальным способом, а сами группы — горизонтальным. Поскольку сигналы, входящие в одну группу, никогда не возникают одновременно, то для их формирования используют дешифраторы.

При вертикально-горизонтальном способе все микрооперации делят на несколько групп. В одну группу включают те микрооперации, которые часто встречаются вместе в одном такте. Эти микрооперации кодируются горизонтально. В каждой микрокоманде предусматривают поле, указывающее на принадлежность микрооперации к определенной группе, формирующее управляющие сигналы на демультиплексоры. Сигналы управления формируются демультиплексорами и подаются на различные схемы процессора.

Для уменьшения объемов управляющей памяти в некоторых компьютерах можно использовать двухуровневое кодирование микроопераций — вертикальным способом кодируется микрокоманда (она находится в памяти микрокоманд), содержащая адрес

памяти нанокоманд, где находится горизонтальная нанокоманда. Сигналы управления формируются содержимым полей нанокоманды. Однако из-за невысокого быстродействия (для формирования одного сигнала управления нужно два обращения в микрокомандную и нанокомандную память) использование такого двухуровневого кодирования микроопераций ограничено компьютерами сравнительно невысокого быстродействия.

Последовательность выполнения микрокоманд. Текущая выполняемая микрокоманда зависит как от предыдущей микрокоманды, так и от осведомительных сигналов, поэтому микропрограмма не может быть сугубо последовательной. В ней обязательно присутствуют безусловные и условные переходы.

Каждой команде компьютера соответствует своя микропрограмма, а следовательно, ее адрес в микропрограммной памяти должен определяться кодом операции. Преобразование кода операции в адрес микропрограммы производится шифратором (см. рис. 5.9), который обычно выполняют в виде специальной постоянной памяти. Дальнейшая очередность выборки микрокоманд определяется принудительным заданием адреса следующей микрокоманды или автоматическим увеличением текущего адреса на единицу (естественная адресация).

Принудительная адресация состоит в том, что в определенном поле каждой микрокоманды содержится информация об адресе следующей микрокоманды. Этот адрес может быть окончательным, не зависящим от значений осведомительных сигналов, или находиться из условия, определяемого в зависимости от таких осведомительных сигналов. Для учета таких условий в микрокоманде предусматривают поле условия перехода. Оно формирует номер осведомительного сигнала, значение которого анализируется при формировании исполнительного адреса следующей микрокоманды. Обычно, если это поле содержит нуль, то условия не проверяются и адрес следующей микрокоманды определяется только адресной частью текущей микрокоманды. Если это поле содержит единицу, то выполняется условный переход к микрокоманде с адресом $A_{МК} = A + X_i$, где X_i — указывает на наличие или отсутствие соответствующего осведомительного сигнала.

Принудительную адресацию микрокоманд отличает высокое быстродействие и универсальность, но она требует значительной емкости микропрограммной памяти, так как каждая микрокоманда содержит адресную часть.

При *естественной* адресации адрес следующей микрокоманды всегда на единицу больше адреса текущей микрокоманды. При последовательном характере микропрограммы отпадает необходимость в наличии адресной части в микрокоманде. При естественном порядке следования микрокоманд для вызова очередной микрокоманды из памяти нужен простой счетчик. Но для организа-

ции переходов в микропрограмму должны быть добавлены специальные микрокоманды условного и безусловного переходов.

В последнее время в качестве устройств микропрограммной памяти все чаще используют флэш-память, допускающую запись новой и изменение хранящейся в ней информации. Этим достигается возможность «настройки» параметров. Загружаемая микропрограммная память позволяет расширять или изменять систему команд, что необходимо при изготовлении проблемно-ориентированных или специализированных компьютеров.

5.7. Методы и средства повышения производительности процессоров персональных компьютеров

Принципы, на которых строится работа процессора, предполагают строго последовательное выполнение всех операций. Но такой процессор не может обладать большой производительностью из-за последовательного характера выполнения операций, поэтому в современных компьютерах используют множество приемов, призванных повысить производительность компьютеров — от использования средств низкоуровневого параллелизма до построения многопроцессорных и многомашинных систем. В этом подразделе рассмотрим некоторые наиболее распространенные методы повышения производительности процессоров, находящие применение в персональных компьютерах. К числу этих методов относятся:

конвейерная обработка;
суперскалярная обработка;
переименование регистров;
динамическое прогнозирование условных переходов;
сопроцессирование.

Конвейерная обработка информации. Конвейеризация команд — один из признанных принципов повышения производительности компьютера. Он находит применение не только в персональных компьютерах, но и в ЭВМ общего назначения. Для выполнения любой команды необходимо устройство, осуществляющее действия в соответствии с ее кодом операции. Для ускорения обработки это устройство выполняют в виде нескольких ступеней, или фаз, причем на каждой ступени производится лишь определенная часть обработки команды. Это так называемый *метод конвейерной обработки команд*: в каждый момент времени на конвейерном процессоре обрабатывается несколько команд, причем все они находятся на разных стадиях обработки. Приход очередного тактового импульса приводит к перемещению обрабатываемых команд на следующую ступень конвейера, полностью выполненная команда

покидает конвейер, а на освободившуюся первую ступень подается новая команда.

Конвейер команд состоит из множества ступеней (так конвейер в одной из последних моделей процессора Pentium IV состоит из 31 ступеней, а в процессоре AMD — из 14), однако для наглядности и простоты изложения мы рассмотрим работу синхронного конвейера, состоящего из пяти ступеней. Такие конвейеры были характерны для многих процессоров персональных компьютеров недалекого прошлого.

Выполнение типичной арифметической команды в пятиступенчатом конвейере можно разделить на пять этапов:

1) выборка команды — код команды заносится в регистр команд;

2) декодирование кода операции — формируются управляющие сигналы;

3) выборка операндов (из регистров) — выбираются операнды, участвующие в выполнении операции и передаются в регистры процессора;

4) собственно выполнение операции — выполнение операции в соответствии с ее кодом;

5) запоминание результата — готовый результат заносится в память или регистры.

При последовательном выполнении команд для реализации программы, состоящей из k команд, потребуется $5k$ тактов, а если их выполнять в конвейере, то всего $(5 + k)$ тактов. Таким образом, пятиступенчатый конвейер обеспечивает ускорение при выполнении программы в $5k/(5 + k)$ раз, но это происходит только при его полной загрузке.

Любые команды в таком конвейере выполняются за пять тактов, но результаты очередной команды появляются в каждом такте, если не учитывать время загрузки и выполнения команды переходов. Результат любой команды будет сформирован только на последней пятой ступени конвейера, т. е. через $5T$ (здесь T означает период между приходом тактовых сигналов) после загрузки команды в конвейер. За это время в конвейер поступит еще четыре команды.

Если первая команда является командой перехода, то следующие четыре уже загруженные в конвейер и начавшие выполнение команды могут оказаться ненужными. Команда перехода нарушает линейный порядок следования команд и требует перезагрузки конвейера. Таким образом, команды переходов снижают эффективность обработки, приводя к приостановкам конвейера. Чем больше команд перехода, тем чаще возникает перезагрузка конвейера и тем меньше производительность.

Пусть m — число ступеней командного конвейера, а n — среднее число команд между двумя соседними командами переходов

(средняя длина линейного участка программы). Выполнение этого линейного участка с помощью конвейера команд завершится за время

$$T_к = T_з + T_о,$$

где $T_з$ — длительность загрузки конвейера; $T_о$ — длительность обработки.

Считая, что длительность загрузки завершается за один такт, а линейный участок программы выполняется в конвейере за n/m тактов, эффективность конвейера (отношение времени выполнения программы в конвейере ко времени последовательного выполнения той же программы $T_п$ в обычном процессоре) можно оценить как

$$T_к / T_п = (1/n + 1/m).$$

На каждой ступени такого конвейера выполняемые действия завершаются за разное время, поэтому подготовленные на одной из ступеней результаты для поддержания синхронного принципа работы должны ожидать завершения обработки на другой. Следовательно, тактовая частота работы конвейера выбирается так, чтобы завершить выполнение действий на самой продолжительной фазе операции. По этой причине, а также из-за необходимости передавать частичные результаты между различными ступенями конвейера его предельное ускорение будет несколько меньше числа фаз.

Построив длинный конвейер, состоящий из множества ступеней, можно еще больше увеличить быстродействие процессора. В длинном конвейере сокращается число вентилей на каждой ступени, поскольку она становится проще, и повышается тактовая частота. Каждая команда разбивается на большее число более коротких фаз, что приводит к незначительному увеличению времени ожидания результатов первой команды, но сокращает время ожидания результатов очередной команды. Однако в таких длинных конвейерах, если приостанавливается выполнение какой-либо команды, то все последующие команды также приостанавливаются; кроме того, не всякую операцию можно разбить на достаточно большое число независимых фаз. Это вызывает возникновение «пустых» фаз, на которых обработка команды не производится, что значительно снижает производительность такого конвейера.

Суперскалярная обработка. Процессор называют *скалярным*, если одновременно на ступень декодирования может подаваться лишь по одной команде, а в процессоре предусмотрен единственный конвейер команд. Однако попытки увеличить производительность привели к тому, что в современных микропроцессорах обычно имеется несколько (от двух до четырех) конвейеров, что позволяет производить одновременную обработку большего числа команд. Такие процессоры принято называть *суперскалярными*.

В зависимости от числа используемых конвейеров современные суперскалярные процессоры бывают двух- и четырехпотоковыми. Эти названия определяет число потоков одновременно обрабатываемых команд. На рис. 5.10 показана структурная схема наиболее простого классического двухпотокового суперскалярного процессора. Команды из кэш-памяти поступают в буферные регистры, откуда передаются на ступень декодирования кода операции. Дальнейшее выполнение команды происходит на конвейере, который в момент декодирования кода операции оказался свободным. В современных процессорах, предназначенных для персональных компьютеров, эти конвейеры обычно неравноценны, поэтому команда для дальнейшего выполнения передается на тот конвейер, который способен ее выполнить.

Ни в одном из суперскалярных процессоров одновременно не может обрабатываться максимальное число команд, на которое он рассчитан, из-за возможных «заторов». Если по какой-либо причине команда в одном из конвейеров не может выполняться, т.е. возникает «затор», то приостанавливается и выполнение команды в другом конвейере, поскольку все команды должны завершать свое выполнение в строго определенном порядке. Это упрощает структуру суперскалярного процессора, но не эффек-

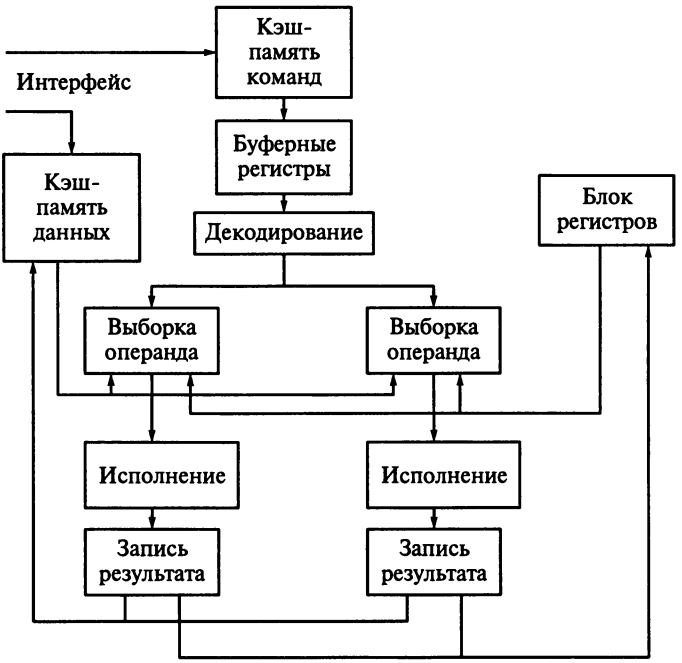


Рис. 5.10. Двухпотоковый суперскалярный процессор

тивно с точки зрения его производительности, так как производительность определяется средним числом команд, обрабатываемых за один такт, а число команд уменьшается из-за простоев конвейера.

Процессоры, допускающие работу одного из конвейеров при «заторе» в других, называют процессорами, обладающими архитектурой с *неупорядоченной обработкой*. Если разрешить выполнение команд в одном из конвейеров при возникновении «затора» в другом, то более поздние команды программы могут оказаться выполненными раньше предшествующих им. Для исключения неправильных результатов нужно, чтобы результаты этих операций не заносились в память, а содержимое регистров не модифицировалось в ошибочной последовательности, т.е. должны быть предусмотрены схемы и методы, позволяющие восстановить правильность вычислительного процесса.

Для реализации неупорядоченной обработки между ступенями декодирования и исполнения в конвейере размещают специальную буферную память, или накопитель команд. Если процессор определяет, что текущая команда не может выполняться, то он забирает из накопителя следующую команду и посылает ее в исполнительный блок. Этот накопитель заполняется командами последовательно, однако порядок формирования результатов может оказаться нарушенным, так как команды имеют разную длительность выполнения, а их исполнение может даже откладываться.

Часто задержки происходят из-за недостаточности аппаратных ресурсов. Например, в компьютере может быть предусмотрен только один порт записи в регистры, а при определенных обстоятельствах конвейеру может потребоваться выполнить две записи в регистровый файл одновременно. Это приводит к конфликту, в результате которого происходит приостановка выполнения команд в одном из конвейеров до тех пор, пока требуемое устройство (один из портов) не станет доступным. Подобную ситуацию называют конвейерным «пузырем» — он проходит по конвейеру, не вызывая никакой обработки, но занимает место в конвейере, а следовательно, снижается производительность обработки.

При наличии в компьютере нескольких конвейеров результаты даже последовательных команд могут формироваться в неупорядоченной последовательности. Пусть нужно выполнить две совершенно независимые команды умножения и сложения. Команда умножения попадает на конвейер первой. Вслед за ней на второй конвейер попадает команда сложения, но ее результат будет получен раньше, так как умножение требует нескольких тактов, а команда сложения одного. Следовательно, имеет место *неупорядоченное завершение* команд.

Если процессор обладает несколькими конвейерами, то неупорядоченное завершение команд неизбежно. При этом резуль-

тат более поздней, но короткой операции может появиться раньше и «занять» регистр, предназначенный для результата предыдущей более длинной команды. Кроме того, необходимо обеспечить получение такого же результата, как и при строго последовательном выполнении команд. Эти проблемы обычно решаются на этапе компиляции программы, но для их решения могут быть использованы и специальные приемы — переименование регистров и переупорядочивание команд.

Переименование регистров. Сколько бы команд одновременно ни был способен выполнять процессор, т.е. сколько бы в нем ни было конвейеров, он никогда не будет загружен полностью. Это связано с тем, что некоторые операции просто нельзя выполнить до получения результатов предыдущих. Например, если надо вычислить значение выражения $(b/2 - d)$, то одновременно произвести операции деления и вычитания не удастся, так как результат операции деления (из которого производится вычитание) неизвестен заранее. Такая ситуация носит название *истинной взаимозависимости* данных и означает, что входные данные для операции вычитания определяются в результате другой операции, в этом случае операции деления. Повысить производительность обработки при истинных взаимозависимостях данных путем одновременного исполнения двух операций невозможно.

Но очень часто в программах существуют *ложные взаимозависимости.* Они возникают в случае, когда следующие друг за другом команды заносят свои результаты в один и тот же регистр. Если команды выполняются не по порядку, то нельзя гарантировать правильность модификации содержимого регистра.

Пусть необходимо выполнить три команды: первая команда имеет большую длительность исполнения (например, команда умножения), вторая команда сложения или сдвига — короткая, а третья команда тоже длинная. В качестве исходных данных вторая команда должна использовать результат, получаемый от первой команды. Если команды выполняются на нескольких конвейерах неупорядоченно, то возможен случай, когда вторая команда (в этом случае сложения) будет завершена до окончания первой, т.е. она использовала прежнее содержимое регистра, в который должен быть помещен результат выполнения первой команды.

Еще один тип ошибок вследствие ложных взаимозависимостей возникает в случае, когда вторая команда может испортить данные, используемые в качестве входных для первой. Этот тип ошибок также возможен при нарушении естественного порядка следования команд. Эти типы ошибок особенно часто встречаются в архитектурах с ограниченным числом РОН.

Для устранения ошибок при ложных взаимозависимостях необходимо воспользоваться *методом переименовывания регистров.* Процессоры, реализующие этот принцип, обладают бóльшим

числом физических регистров, чем определяется архитектурой. Увеличить число архитектурных регистров невозможно, так как это привело бы к нарушению программной совместимости компьютеров. Если какой-либо команде требуется регистр, то процессор динамически ставит в соответствие (переименовывает) этому архитектурному регистру один из нескольких свободных физических регистров. Когда другая команда пытается обратиться к тому же архитектурному регистру, ему ставится в соответствие уже другой физический регистр.

Для динамической подстановки номера физического регистра вместо логического служит справочная таблица, которая обновляется после декодирования каждой команды. Результат заносится в свободный физический регистр, но адрес логического регистра запоминается для восстановления вычислительного процесса при прерываниях. Естественно, что переименование не может быть «вечным», оно действует только, пока команды продвигаются по конвейерам.

Возможна и другая организация процедуры переименования, для которой используются набор регистров и буфер переименования. В каждом регистре набора помимо значения операнда предусмотрен дополнительный разряд, указывающий на достоверность этого значения, а буфер переименования выполнен в виде регистров и построен по ассоциативному принципу. Каждый регистр буфера переименования содержит поля для указания доступности или занятости регистра, номера переименованного регистра, текущего значения содержимого регистра, достоверности и последнего переименования. Эти поля служат для нахождения свободного регистра в буфере переименования, записи в него результата и нахождения значения операнда при операциях чтения. Переименование регистров не исключает влияния истинных взаимозависимостей. Однако их влияние можно уменьшить *методом обхода*, при котором результаты выполнения одной команды пересылаются прямо другой в соответствующий конвейер без запоминания их в регистрах или памяти. Это позволяет процессору выполнять некоторые команды одновременно, немедленно передавая результаты одной из них в другую.

Динамическое прогнозирование условных переходов. Конфликты по управлению могут привести к существенным потерям производительности конвейера, значительно превышающим потери от конфликтов по данным. Такой конфликт может возникнуть, если выполняется команда условного перехода. *Переход* — это изменение последовательности выполнения команд; он может зависеть от результата предыдущей операции (например, операции сравнения). При реализации команд переходов используется специальный регистр — регистр флагов, куда заносятся указатели, или флаги, характеризующие результат любой операции. В персо-

нальных компьютерах IBM используются флаги переноса, четности, переноса из младшей тетрады в старшую, нуля, знака, переполнения, направления обработки цепочек и разрешения прерываний. Переходы встречаются в программе довольно часто. Например, в программах для персональных компьютеров команды переходов приходятся в среднем на каждые шесть-семь выполняемых команд.

Адрес условного перехода становится известным только на исполнительной ступени конвейера, т.е. до завершения операции перехода процессор просто не знает, какую команду ему следует направить в конвейер после команды перехода. Так, если в результате выполнения предыдущей операции ее результат меньше нуля, то процессор должен выполнять одну команду, но если результат больше нуля, то другую, хранящуюся в совершенно иной ячейке памяти. Однако о направлении перехода будет известно только после выполнения предыдущей команды.

Для выполнения перехода нужно сделать предположение о пути ветвления. Такой метод получил название *прогнозирования ветвления*. Если бы им не пользовались, то при возникновении перехода процессору пришлось бы выбрасывать частично выполненные, но совершенно ненужные команды. Особенно большие потери такое выбрасывание вызывает в суперскалярных процессорах, поскольку в них одновременно обрабатывается большое число команд.

Процессор может спрогнозировать переход и начать выборку команды из ячейки с предсказанным адресом задолго до того, как узнает о правильности своего прогноза или до получения условия перехода, однако не может изменять содержимое архитектурных регистров или ячеек памяти. В некоторых процессорах предусмотрены специальные средства, обеспечивающие несколько уровней предположений.

Если процессор неверно спрогнозировал переход (это выяснится только после завершения выполнения команды условного перехода), он должен отменить все полностью или частично выполненные команды и очистить конвейеры. Неверно предсказанные переходы очень сильно снижают производительность, поэтому во всех ЦП принимают меры к снижению их количества.

Пусть на этапе выборки определено, что надо выполнить команду перехода. Тогда для уменьшения потерь времени нужно знать, по какому адресу выбирать следующую команду. Однако до дешифрирования неизвестно, что это за команда и чему равно следующее значение счетчика команд. Если сохранить в специальном буфере адрес команды перехода, прогнозируемый адрес следующей команды и предысторию переходов, то можно сократить число неверных переходов. На этапе выборки очередной команды производится обращение к этой буферной памяти, для чего используется значение счетчика команд (рис. 5.11). При обнаруже-

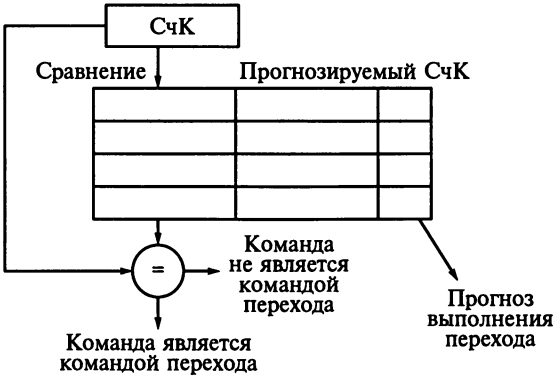


Рис. 5.11. Использование специального буфера для организации переходов

нии совпадения прогнозируются условия перехода и немедленно производится выборка и дешифрация следующей команды.

Этот метод применяется в персональных компьютерах Pentium, где данные о последних переходах хранятся в специальном буфере. Переход осуществляется по адресу, сохраненному в этом буфере.

Можно добавить информацию о переходе к каждой строке кэш-памяти команд и указать, как следует выполнять ветвление (как выполняемое или невыполняемое). Подобный принцип нашел применение в процессорах фирмы AMD.

Сопроцессирование. *Сопроцессор* — это дополнительное операционное устройство, работающее под управлением основного процессора и призванное выполнять ряд команд, которые по тем или иным причинам не могут быть реализованы в основном процессоре. Так, добавление сопроцессора операций с ПТ (а это наиболее распространенный вид сопроцессоров) позволяет увеличить скорость выполнения арифметических операций в десятки и сотни раз. При этом сложность схем сопроцессора соизмерима со сложностью основного процессора.

Процессор и сопроцессор используют одну и ту же память, но ее работой управляет только основной процессор. Счетчик команд и средства определения исполнительных адресов находятся в основном процессоре, а в сопроцессоре обычно размещаются собственные регистры, дешифратор команд и АЛУ. Основной процессор реализует набор команд IS_M, а сопроцессор — дополнительный набор команд IS_C. На рис. 5.12, *а* показано устройство обработки, состоящее из основного и дополнительного процессоров, реализующее систему команд $IS = IS_M \cup IS_C$. Все управление памятью сосредоточено в основном процессоре; все команды реализуются последовательно. Значительное повышение скорости работы системы достигается за счет сокращения длительности

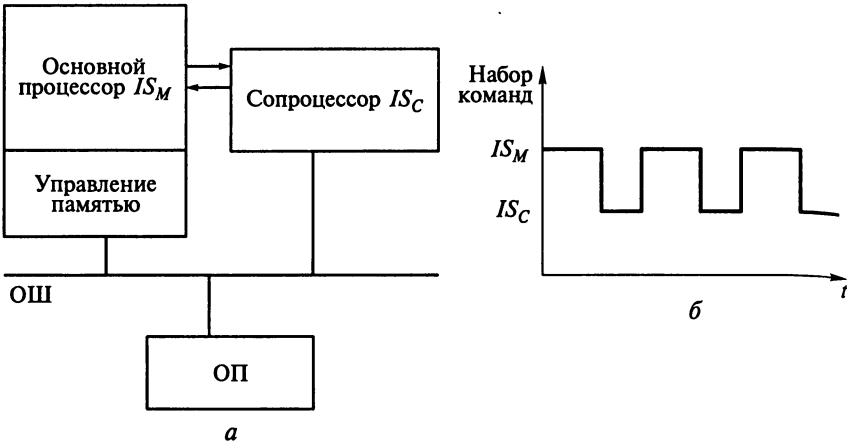


Рис. 5.12. Последовательная работа процессора и сопроцессора:
а — функциональная схема; *б* — длительность выполнения операций

выполнения операций IS_C, которые в отсутствие сопроцессора реализуются в виде подпрограмм (рис. 5.12, *б*).

Содержимое счетчика команд из основного процессора передается в ОП, откуда команда одновременно заносится на регистры команд как основного, так и дополнительного процессоров. Производится декодирование кода операции и формирование сигналов управления в том процессоре, системе команд которого соответствует декодируемая операция. После завершения выполнения операции в любом из процессоров формируется сигнал разрешения выборки следующей команды.

Чтобы добиться еще большего повышения быстродействия, работу основного и дополнительного процессоров совмещают во времени. Однако в этом случае организация работы усложняется. Выбранная команда вначале попадает в буфер, организованный по принципу FIFO, из которого и передается на обработку в процессор или сопроцессор. Для того чтобы такая передача состоялась, нужно чтобы соответствующий процессор оказался свободным. Команды из набора IS_M передаются из буфера на исполнение в основной процессор, а команды из набора IS_C — в сопроцессор. Так как длительность выполнения команд различна, то завершение последующей команды (в основном процессоре) может опережать завершение предыдущей (в сопроцессоре), т. е. может быть нарушена последовательность результатов обработки. Избежать этого можно присвоением командам тегов очередности. Такой тег содержит порядковый номер команды. Этот же тег присваивается и результату, что позволяет поместить результат в соответствующее место выходной очереди. Однако важно, чтобы выполнение

текущей команды не зависело от признаков, вырабатываемых предыдущей. Эта задача возлагается на компилятор.

Структурные схемы процессоров Alpha и Pentium. Наиболее распространены в нашей стране процессоры RISC- и CISC-архитектур. Рассмотрим упрощенные структурные схемы процессоров Alpha и Pentium (рис. 5.13, 5.14). В них реализуется большинство описанных выше принципов, служащих для повышения быстродействия.

Поскольку RISC-архитектура считается одной из наиболее перспективных, то в качестве примера рассмотрим 64-разрядный микропроцессор Alpha 21066 фирмы DEC, в котором реализованы принципы RISC-обработки. Процессоры Alpha отличаются универсальностью, позволяющей применять различные операционные системы, а также хорошей сбалансированностью, т.е. возможностью работы с огромными массивами данных, находящимися на дисках и в ОП. Однако в настоящее время работы по дальнейшему совершенствованию процессоров Alpha прекращены, так как отделение фирмы, занимающееся разработкой этих процессоров, продано.

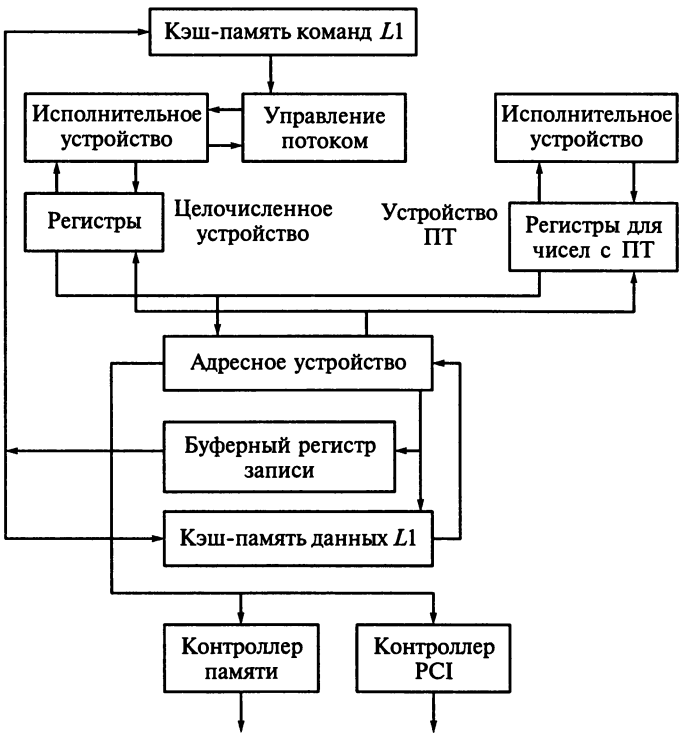


Рис. 5.13. Структурная схема процессора Alpha 21066

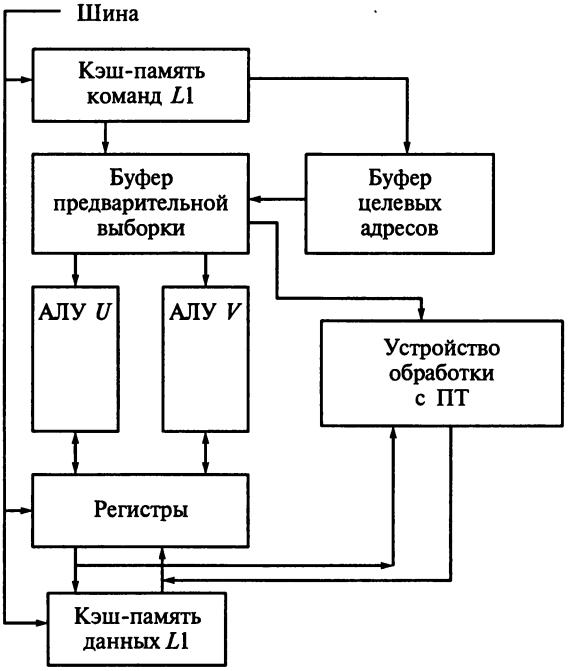


Рис. 5.14. Упрощенная структурная схема процессора Pentium

Структура такого процессора приведена на рис. 5.13. Микропроцессор выполнен на одном кристалле, и в его состав входят устройства целочисленной и плавающей арифметики; кроме того, в нем предусматривается кэш-память уровня $L1$ емкостью 32 Кбайт. Он работает на тактовой частоте 275 МГц; частота более поздних модификаций значительно выше.

В кристалле процессора реализованы конвейерная организация всех функциональных устройств, одновременная выдача нескольких команд и средства симметричной многопроцессорной обработки.

В кристалле процессора Alpha расположены два блока, состоящие из 32 регистров каждый: первый предназначен для хранения целых чисел, а второй — чисел с ПТ. Этот процессор способен выполнять арифметические операции над числами с одинарной и двойной точностью.

Основными компонентами являются кэш-память команд и данных, устройства обработки чисел с ФТ, обработки чисел с ПТ, устройство для выполнения команд загрузки и записи, а также контроллеры памяти и шины PCI.

Кэш-память выполнена как кэш прямого отображения. При выполнении операции записи в память данные одновременно за-

писываются и в кэш, и буферный регистр записи. Контроллер памяти реализует алгоритм отложенного копирования. При обнаружении промаха контроллер обращается к ОП для загрузки соответствующих строк кэш-памяти. Предусмотрен контроллер прямого доступа к шине PCI. К этой шине подключаются адаптеры для сопряжения с шиной прежнего стандарта ISA с целью использования «старых» периферийных устройств (ПУ) и шиной SCSI для подключения дисковых накопителей.

Процессор Alpha имеет две модификации, построенные по технологии RISC; в них предусматривается кэш-память первого уровня объемом 16 Кбайт и второго уровня объемом 96 Кбайт. Для ускорения обменов с памятью в них реализована 128-разрядная шина, позволяющая передавать из памяти два двойных слова, что значительно повышает пропускную способность памяти.

Одной из наиболее интересных RISC-архитектур является многопроцессорная архитектура SPARC (Scalable Processor Architecture) фирмы Sun. Процессоры с такой архитектурой изготавливаются уже несколькими фирмами. В настоящее время с такой архитектурой выпускаются 64-разрядные процессоры. К основным особенностям такой архитектуры относятся:

регистровые окна в качестве механизма передачи параметров между программами;

механизмы отложенных переходов и аннулирования команд;

широкое использование принципов суперскалярной обработки;

многоступенчатые конвейеры целочисленной арифметики и арифметики с ПТ;

отделение подсистемы ввода-вывода от подсистемы процессора.

Рассмотрим структуру современного CISC-процессора на примере наиболее распространенного в нашей стране процессора Pentium IV (см. рис. 5.14). В основу этого микропроцессора положена идея суперскалярной обработки, он обладает следующими особенностями:

двухпотоковая (или четырехпотоковая) суперскалярная организация, допускающая одновременное выполнение двух (или четырех) простых команд;

наличие двух независимых кэш-памятей первого уровня для команд и данных, построенных по множественно-ассоциативному принципу;

динамическое прогнозирование переходов;

конвейерная 8-ступенчатая организация блока для выполнения операций над числами с ПТ.

Все команды распределяются по двум (а в более поздних версиях Pentium IV — по четырем) исполнительным устройствам, реализующим конвейерный принцип обработки. Конвейер *U* спо-

собен выполнять любые команды из системы команд x86, в том числе целочисленной арифметики и над числами с ПТ. Конвейер V предназначен для выполнения простых целочисленных команд. В эти конвейеры команды направляются одновременно, причем более сложная поступает в конвейер *U*, а более простая — в конвейер *V*. Такая одновременная выдача двух команд возможна только для ограниченного подмножества целочисленных команд и только при отсутствии зависимости команд по регистрам. При любой остановке выполнения команды в первом конвейере останавливается и второй конвейер.

В этом процессоре используются раздельные кэш-памяти данных и команд *L*1 емкостью по 16 Кбайт (в последних моделях Pentium IV). За один такт из них может считываться по два слова, записанных в одну строку. В кэш-памяти записываются три тега, позволяющие производить считывание данных из одной или двух смежных строк и следить за когерентностью кэш-памяти. Для ускорения загрузки кэш-памяти служит 64-битовая шина данных. Число адресных линий составляет 32.

Для реализации механизмов динамического прогнозирования переходов в процессоре предусмотрены буфер целевых адресов переходов и два буферных регистра предварительной выборки. В буфер целевых адресов переходов заносятся адреса команд, находящихся в буферных регистрах предварительной выборки. При обнаружении в потоке команд операции перехода указанный в ней адрес сравнивается с адресами, помещенными в буфер целевых адресов.

При их совпадении считается, что данный переход будет выполнен, и начинают выдаваться команды в соответствующий конвейер. Окончательное решение о направлении перехода принимается на основании анализа кода условия. При неправильном прогнозе содержимое конвейеров теряется, что приводит к приостановке их работы на 3—4 такта.

Помимо механизмов, предназначенных для ускорения операций над числами с ФТ и управляющих операций, в этом процессоре осуществляется конвейерная 8-ступенчатая обработка чисел с ПТ.

В последних моделях этого процессора используются механизмы:

выполнения команд с нарушением их последовательности в программе, что во многих случаях сокращает приостановку конвейеров на ожидание операндов;

переименования регистров, что позволяет увеличить число используемых регистров сверх восьми и тем самым ускорить продвижение команд по конвейеру.

Эти механизмы и возможности суперскалярной обработки реализованы во многих процессорах фирмы Intel. Архитектура про-

цессоров Pentium IV оптимизирована для 32-разрядных прикладных программ; это значит, что все внутренние регистры и средства передачи информации рассчитаны на работу с 32-разрядными словами. Для полной реализации всех возможностей таких процессоров требуется 32-разрядная ОП и 32-разрядные прикладные программы. В системах, построенных на основе процессора Pentium IV, передача данных осуществляется по 64-разрядной магистрали, имеющей дополнительно восемь разрядов для использования корректирующих кодов; передача адреса выполняется по 36-разрядной магистрали. В настоящее время процессоры Xeon фирмы Intel, предназначенные для работы в качестве высокопроизводительных серверов сетей, рассчитаны на обработку 64-разрядных данных. Но процессоры Xeon и Pentium IV имеют практически одинаковые процессорные ядра, поэтому в Pentium IV также предусмотрены блоки для обработки 64-бит данных, но пока они отключены.

При выполнении прежних прикладных программ, рассчитанных на 16-разрядную обработку, возникают значительные задержки. В процессоре Pentium IV высокая производительность достигается благодаря методу исполнения с изменением последовательности команд, который обеспечивает максимальную загрузку конвейеров; однако 16-разрядные команды не могут выполняться вне очереди, что превращает их в серьезную помеху, вынуждая конвейерный процессор останавливать выполнение всех других операций до завершения этой команды. Особенно вредное влияние на производительность Pentium IV оказывают команды загрузки сегментных регистров (например, регистров, в которых хранится сегментная часть адреса) и команды ввода-вывода. Они не только задерживают выборку и обработку следующих за ними команд, но и требуют полной очистки всех ступеней конвейера. Это приводит к тому, что для достижения высокой производительности этих процессоров необходимы соответствующие 32-разрядные, а в будущем и 64-разрядные программы.

Быстродействие процессора определяется не только принципиальной схемой, но также технологией и конструкцией, наличием или отсутствием кэш-памяти, ее объемом и скоростью работы. Объем кэш-памяти *L2* может составлять 512 Кбайт, 1 или 2 Мбайт. В современных процессорах имеются расширения системы команд MMX (Multimedia Extensions) для реализации функций мультимедиа и SSE (Streaming SIMD Extensions) для реализации мультипроцессорных конфигураций. Эти команды предназначены для многопроцессорных конфигураций, которые все чаще используются в сетях при построении серверов. Очень немногие обычные пользователи что-либо реально выигрывают от таких расширений, но, во-первых, сегодня машины используются не столько для решения сугубо производственных задач, сколько для игр, и,

во-вторых, современные компьютеры содержат множество иных средств, обеспечивающих более высокую производительность при работе с деловыми программами.

Увеличению скорости работы процессора способствует и уменьшение его размеров, которое сопровождается снижением напряжения питания. Так, питание процессора Xeon осуществляется от 1,8 В, что приводит к снижению потребляемой мощности, а рабочая частота составляет свыше 2,0 ГГц. Тактовая частота процессоров Pentium IV составляет свыше 3,5 ГГц, а напряжение питания — менее 1,5 В. При переходе с технологии 0,13-мкм на 0,09-мкм можно ожидать, что будут достигнуты значения тактовой частоты порядка 4,5 ГГц. Кроме того, повышению частоты в процессорах Pentium IV способствует фактическое объединение двух кристаллов в одном корпусе: собственно процессора и кэш-памяти *L2* объемом 512 Кбайт (или 1 Мбайт).

Контрольные вопросы

1. Что представляет собой процессор? Что такое система команд? Какие команды входят в ее состав?
2. Какие поля образуют команду? Для каких целей и как используется поле кода операции?
3. Сколько адресов может содержать одна команда? Для каких целей они служат?
4. Какие типы команд входят в состав системы команд?
5. Как формируется исполнительный адрес, если используется базовый метод адресации?
6. Что представляет собой микропрограммный автомат с «жесткой» логикой? Где они находят применение?
7. Каковы достоинства и недостатки автоматов с «жесткой» логикой?
8. Перечислите достоинства и недостатки автоматов с программируемой логикой.
9. Какие существуют способы кодирования микрокоманд? Перечислите их достоинства и недостатки.
10. Поясните принцип управления вычислениями на основе микропрограммного автомата с программируемой логикой.
11. В чем заключаются причины появления архитектуры процессора с сокращенным набором команд?
12. Какие особенности отличают RISC-компьютер от его «старшего собрата» CISC-компьютера?
13. Приведите структурную схему RISC-компьютера. Как в нем происходит обработка информации, находящейся в ОП?
14. Какие особенности характерны для CISC-компьютеров? Приведите пример компьютера с архитектурой CISC.
15. Каково назначение конвейера команд? Сколько ступеней используется в современных конвейерных процессорах персональных компьютеров?

16. Что такое многопотоковая суперскалярная обработка? Приведите пример схемы процессора, реализующего такую обработку.

17. Каким образом можно уменьшить время, требуемое для реализации условных переходов? Каково назначение специального буфера для организации переходов?

18. Как выполнить команды с нарушением последовательности? Какие проблемы они порождают?

19. Что достигается с помощью переименования регистров и как оно осуществляется?

20. Приведите структурную схему CISC-компьютера (например, IBM PC). Каково назначение его узлов?

21. Что дает конвейерная организация обработки информации, к каким трудностям она приводит?

22. На какие этапы можно разбить выполнение команды? Как организовать конвейерную обработку?

23. Как и в каких случаях производится перезагрузка конвейера? К чему она приводит?

24. Каковы особенности новых микропроцессоров Pentium IV и AMD? Каким быстродействием они обладают?

25. Что такое истинная и ложная взаимозависимости? Какие существуют средства борьбы с ложной взаимозависимостью?

26. Что такое MMX и SSE?

27. Какими особенностями, характерными для RISC-компьютеров, обладает процессор Pentium IV?

ГЛАВА 6

ОРГАНИЗАЦИЯ ПАМЯТИ

6.1. Общие сведения

Высокой производительности компьютера невозможно добиться без наличия быстродействующей памяти достаточно большого объема. В памяти должны храниться программы обработки данных и сами данные. Для достижения высокой скорости обработки необходимо, чтобы память содержала большие объемы перерабатываемой информации, и чтобы к этой информации обеспечивался быстрый доступ. Память принято характеризовать емкостью, быстродействием и стоимостью хранения информации. Помимо этого важно знать метод доступа и величину пересылаемого кванта информации.

Емкость памяти — это количество информации (бит или байт), которое может храниться в ней. Современный компьютер обладает огромной емкостью памяти, поэтому ее измеряют в более крупных единицах — кило-, мега-, гига- или терабайтах. (Если приставка пишется с большой буквы, то она соответствует целой степени числа 2.)

Быстродействие памяти определяется временем доступа и длительностью цикла. Важной характеристикой служит также скорость передачи. *Время доступа* — это интервал между поступлением запроса на чтение или запись и моментом выдачи запрашиваемых данных, а *длительность цикла* — минимально возможное время между двумя последовательными обращениями к памяти.

Известны четыре метода доступа: последовательный, прямой, произвольный и ассоциативный. Характерным примером устройства с последовательным методом записи может служить запоминающее устройство (ЗУ) на магнитной ленте. Чтобы прочитать требуемую информацию в устройстве с последовательным доступом, нужно прочитать все предшествующие записи. Время доступа зависит от места расположения записи на носителе. Прямой доступ характерен для НМД. Для поиска нужной информации вначале магнитная головка перемещается к началу записи, а затем производится последовательное чтение данных из этой записи. При произвольном доступе каждое слово хранится в отдельной ячейке с уникальным адресом. Время обращения к любой ячейке

постоянно. Примером ЗУ с произвольным доступом служит основная память. В ЗУ с ассоциативным доступом поиск нужной информации осуществляется путем сравнения отдельных битов каждого хранящегося в памяти слова с образцом. По ассоциативному принципу построена кэш-память.

Современные интегральные схемы самой быстрой статической памяти имеют цикл порядка 5... 12 нс (а их емкость составляет 2^6 байт). Это в 10 раз больше задержек на переключение вентилей в логических схемах. Кроме того, статическая память очень дорогая, а цикл более дешевой динамической памяти составляет около 50 нс, что уже в 100 раз больше времени переключения вентилей процессора. И статическая, и динамическая полупроводниковые памяти могут хранить информацию только при наличии питания, поэтому при выключении компьютера информация в них теряется.

Все это привело к тому, что память имеет иерархическую структуру, в которой часто используемая процессором информация переносится на более высокий уровень иерархии. При многоуровневой организации памяти современного компьютера (рис. 6.1) память включает в себя следующие уровни: сверхоперативный (регистры, кэш-память), оперативный (ОП, состоящая из оперативной и постоянной), внешний (образуемый НМД) и архивный.

Емкость памяти на каждом более удаленном от процессора уровне увеличивается, а быстродействие снижается. Кроме того, уменьшается стоимость хранения информации по мере «удаления» уровня от процессора. Однако наличие нескольких уровней памяти не приводит к увеличению общего объема хранимой информации — на каждом уровне информация дублируется той, что находится на последующем более низком уровне.

Процессор производит непосредственную обработку только той части информации, которая находится на первых двух уровнях — оперативном и сверхоперативном. Система управления памятью осуществляет обмен информационными блоками между уровнями памяти, причем при обращении к блоку, находящемуся на низком уровне, он перемещается на более высокий уровень. Это делается для того, чтобы при последующих обращениях к этому же блоку проводить выборку из запоминающего устройства, обладающего большим быстродействием.

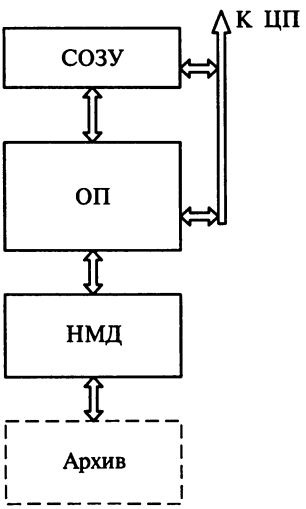


Рис. 6.1. Многоуровневая организация памяти

Рис. 6.2. Карта адресов памяти персонального компьютера

Если обращение к какому-либо уровню памяти прошло успешно, т.е. в нем найден нужный информационный блок, то говорят о попадании (hit), а когда блок на этом уровне отсутствует — о промахе (miss). В случае промаха для поиска нужного блока необходимо обратиться к следующему уровню памяти. Потери времени при этом включают в себя время на поиск и пересылку искомого слова из следующего более медленного уровня памяти.

Помимо перечисленных видов памяти, в компьютере имеется управляющая память, предназначенная для размещения микропрограмм процессора. Это постоянная память, в которой хранятся управляющие программы, константы и т.д. Кроме того, другая постоянная память содержит программы начальной загрузки.

На каждом уровне запись информации выполняется по своим законам. Так, во внешней памяти на дисках запись производится по секторам, содержащим как собственно сохраняемую информацию, так и вспомогательную, необходимую для поиска нужных данных. В ОП каждое слово записывается в отдельную ячейку, а для того чтобы найти и воспользоваться им, нужен адрес этой ячейки.

В сверхоперативной памяти слово записывается в ячейку вместе с полным адресом ячейки ОП, где оно хранилось или должно храниться, или его частью.

Основная память машины является адресуемой; это значит, что вся ОП представляет собой множество ячеек, отличающихся друг от друга только адресами. Если процессору нужно воспользоваться каким-либо словом, он должен передать в память его адрес. Все множество адресов образует *адресное пространство*, но не все адреса принадлежат ОП. Эти адреса могут использоваться для обращения:

в постоянную память;
оперативную память;
память дисплея (видеопамять);
к регистрам периферийных устройств.

Распределение исполнительных адресов, а следовательно, и объемов памяти по конкретным устройствам и образует адресное пространство (рис. 6.2).

6.2. Виды памяти

Постоянная память. Обычно постоянная память служит для хранения программ начальной загрузки и ряда других системных программ. Как правило, ячейкам ПЗУ присваиваются адреса адресного пространства вблизи 1 М (см. рис. 6.2).

Постоянное запоминающее устройство выполняет преобразование *m*-разрядного кода адреса в *n*-разрядный код хранящегося по этому адресу слова. Таким образом, ПЗУ можно рассматривать в качестве преобразователя кодов. Оно значительно проще оперативной памяти (информация не может записываться в ПЗУ при обычном режиме работы), надежнее и дешевле ее, но обычно медленнее.

Информация в ПЗУ записывается прямо на кристалле при нанесении маскирующего слоя и становится неотъемлемой частью этого кристалла. Это не позволяет менять информацию. Более гибкий подход появился с созданием программируемого ПЗУ (ППЗУ). Для процесса записи информации на таких кристаллах требуется более высокое напряжение, чем при записи в ОЗУ. Это приводит к «прожиганию» перемычек в различных местах памяти и, следовательно, записи информации на этом кристалле. Запись в ППЗУ производится на специальных устройствах, называемых *программаторами*, после чего интегральная схема ППЗУ устанавливается в компьютер. После записи изменить записанные данные в компьютере уже невозможно.

В перепрограммируемом ПЗУ можно записать информацию в память, а при необходимости внести в нее изменения, можно всю информацию стереть и начать повторную запись. Запись информации в перепрограммируемое ПЗУ также выполняется на программаторах.

Последним шагом стало создание электрически стираемых программируемых постоянных запоминающих устройств (ЭСППЗУ), напоминающих ППЗУ, но обладающих одним преимуществом: стирание информации в них может производиться просто сигналом более высокого напряжения.

В настоящее время большое распространение получила флэш-память, представляющая собой особый вид ЭСППЗУ, в котором стирание информации осуществляется блоками или целиком всего содержимого памяти; при этом используются сигналы обычного для компьютера напряжения, но большой длительности. Запись байта информации во флэш-память занимает около 10 мкс и производится сигналами, подаваемыми на адресные входы и информационные выходы (которые в режиме записи играют роль информационных входов). Флэш-память используют не только в качестве памяти BIOS, но и для цифровой записи фотоснимков и других целей.

Для хранения программ BIOS в настоящее время используют ПЗУ на МОП-элементах и флэш-память.

Оперативная память. Оперативное запоминающее устройство является основной частью адресуемой памяти компьютера, поэтому его часто называют основной памятью. За последние годы ее объем увеличился в сотни раз и в настоящее время в персональных компьютерах достигает 4 Гбайт (он определяется длиной адресной части команды, которая составляет 32 разряда), а в серверах может быть и значительно больше. Информация, постоянно хранящаяся во внешних ЗУ, становится доступной процессору только после загрузки ее в ОП. Оперативная память — это ЗУ с произвольным доступом, длительность записи и чтения постоянны и не зависят от номера ячейки.

Основную память можно рассматривать в виде линейной последовательности одинаковых ячеек (обычно в современных компьютерах их длина составляет один байт). При обращении к ОП передается адрес ячейки (по адресной шине), данные (по информационной шине) и управляющие сигналы (например, сигналы записи или чтения). Конструкция компьютера может привести к необходимости совмещения адресной и информационной шин, тогда по такой совмещенной шине адреса и данные передаются последовательно.

В настоящее время оперативная память представляет собой энергозависимое устройство, т. е. информация может храниться в нем только при наличии питания; при выключении питания она не сохраняется. (В качестве ОП в компьютерах второго и третьего поколений использовали сложную в производстве дорогостоящую ферритовую память, но теперь, когда применение компьютеров стало повсеместным, ей пришла на смену более технологичная полупроводниковая память. Ферритовая память обладает рядом достоинств, она энергонезависима, т. е. сохраняет записанную в ней информацию при выключении питания, и обладает радиационной стойкостью.)

Основная память, как правило, строится на сравнительно недорогих динамических элементах, но ее быстродействие значительно отстает от быстродействия процессора и растет очень медленно. Время записи или чтения информационного слова из такой памяти существенно превышает длительность цикла процессора и в настоящее время составляет порядка 30...50 нс.

Запоминающий элемент динамической памяти состоит из конденсатора *C*, отсутствие или наличие заряда в котором и означает «0» или «1», а также транзистора, служащего для изоляции конденсатора от остальных схем. Кроме того, для записи и чтения предусматривают еще два транзистора (рис. 6.3). Размеры конденсатора малы, это позволяет размещать запоминающие элементы в кристалле с высокой плотностью. Но малые размеры конденсато-

ра означают, что он обладает и очень небольшой емкостью, следовательно, время утечки заряда, т.е. длительность хранения информации в нем, также незначительно и составляет десятки миллисекунд. Поэтому динамические элементы требуют для постоянного поддержания в них информации специальных циклов регенерации, которая производится каждые 2...8 мс. *Регенерация* — это процесс принудительного чтения и записи считанной информации в ту же ячейку памяти. Если содержимое ячейки использовалось в течение времени между циклами регенерации, то процесс регенерации этой ячейки можно опустить.

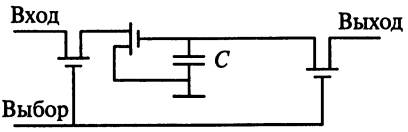


Рис. 6.3. Запоминающий элемент динамической памяти

Оперативную память можно представить в виде матрицы запоминающих ячеек; тогда для обращения к ячейке, т.е. для записи или чтения данных из нее, нужно передать адрес $A_n...A_0$ этой ячейки. Адрес включает в себя две составляющие: адрес строки (raw) и адрес столбца (column). Они поступают на регистр адреса ОП (обычно одновременно) и затем дешифрируются. К выходам дешифраторов подключены горизонтальные и вертикальные адресные линии матрицы ячеек, образующие соответственно строку и столбец матрицы. Помимо адресных линий все ячейки связаны вертикальными линиями, по которым данные $D_m...D_0$ поступают в ОП или считываются из нее. Данные, подлежащие записи в память, поступают на информационный регистр, а затем на выбранную ячейку. При операциях чтения данные из этой ячейки поступают на тот же информационный регистр, а затем передаются по шине в процессор.

Управление ОП осуществляется сигналами разрешения записи (WE), разрешения выдачи выходных сигналов (OE) и выбора микросхемы памяти (CS); помимо этого адрес строки и столбца сопровождается стробирующими сигналами RAS и CAS (рис. 6.4).

Контроллер памяти, т.е. схема, вырабатывающая сигнал управления при обращении к ОЗУ, обычно строится по синхронной схеме. Каждая операция в памяти требует не менее пяти тактов: определение типа операции (чтение или запись) и установка адреса строки, формирование сигнала RAS, установка адреса столбца, формирование сигнала CAS, перевод сигналов RAS и CAS в первоначальное состояние. Поскольку адрес строки и столбца используются не одновременно, то для уменьшения числа контактов микросхемы памяти в большинстве случаев производится их мультиплексирование.

Быстродействие компьютера неразрывно связано с повышением пропускной способности памяти, что можно достичь несколькими способами. Во-первых, между процессором и ОП можно

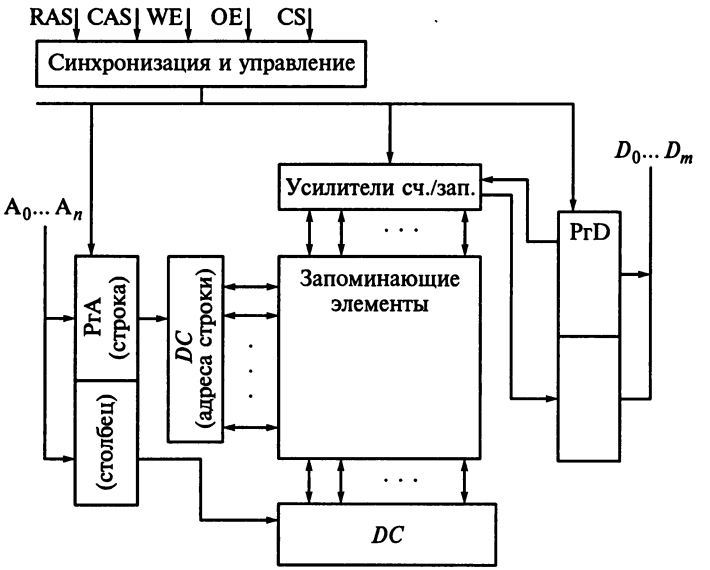


Рис. 6.4. Структура ОП

установить более быструю буферную память (кэш-память), куда помещать обрабатываемые данные. Во-вторых, выполнять расслоение памяти, т.е. строить ОП из нескольких блоков и размещать в них данные так, чтобы обращение к ним производилось последовательно. В-третьих, производить выборку из ОП широким слоем и расформировывать его на отдельные команды или слова данных в регистрах процессора.

Если память строится в виде нескольких модулей с автономными схемами адресации, записи и чтения, то можно организовать расслоение памяти. Под *расслоением* понимают такую организацию ОП, когда ее выполняют из нескольких автономных модулей, в которые информация заносится по определенным правилам.

Известно несколько способов организации расслоения. В персональных компьютерах для этих целей чаще всего используется разделение памяти на память данных и память программ. Такое расслоение применяется при организации внутренней кэш-памяти практически всех современных микропроцессоров. Сегодня в каждом микропроцессоре имеется внутренняя кэш-память объемом от 8 до 64 Кбайт и более, разделенная на два блока — память команд и память данных. При выполнении любой команды требуется несколько обращений в память как за самой командой, так и операндами; эти обращения производятся в разные блоки внутренней памяти.

В случае классического расслоения памяти ее строят из нескольких модулей; ячейки с последовательными адресами находятся в различных модулях. Благодаря свойству локальности программ (и данных) вслед за текущей командой вероятнее всего следующей будет выполняться команда, адрес которой на единицу больше адреса текущей команды, т.е. команда, хранящаяся в следующем модуле памяти. Такой порядок нарушается только командами переходов. Данные также записываются в память и используются программой последовательно, т.е. из разных модулей. Такое расслоение возможно только при постоянном формате команд.

Расслоение памяти чаще всего организуют в соответствии с младшими разрядами. Адрес включает в себя две составляющие (рис. 6.5, *а*):

$$A = A_{ст}, A_{мл},$$

где $A_{ст}$, $A_{мл}$ — старшие и младшие разряды соответственно.

Все программы и данные располагаются в памяти последовательно. Однако ячейки со смежными адресами находятся в разных физических блоках.

Предположим, что в машине используются четыре модуля памяти, а номер модуля соответствует содержимому двух младших разрядов адреса $A_{мл}$. Тогда все обращения к ячейкам памяти с последовательными адресами будут приходиться на разные модули, а

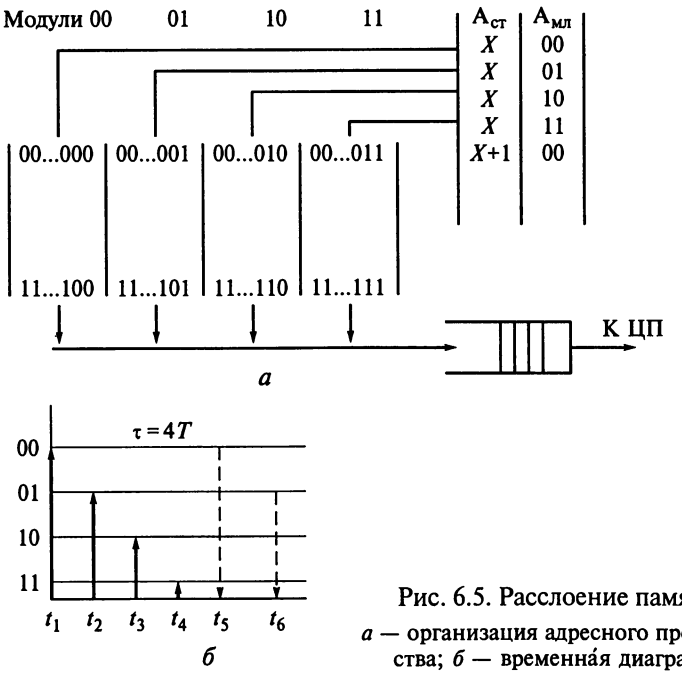


Рис. 6.5. Расслоение памяти:
а — организация адресного пространства; *б* — временна́я диаграмма

поскольку все эти модули обладают собственными схемами адресации и выборки, то можно производить обращение к следующему модулю, не дожидаясь ответа от предыдущего.

На временно́й диаграмме (рис. 6.5, *б*) время обращения и выборки из каждого модуля τ составляет четыре такта T ($\tau = 4T$). Обращения к памяти происходят непрерывно в моменты t_1, t_2, t_3 и т.д. Последовательные обращения приходятся на разные модули, поэтому суммарный темп выдачи квантов информации из памяти соответствует одному такту T, а выдача квантов информации из каждого отдельного блока производится с темпом $4T$. Задержка в выдаче кванта информации относительно момента обращения к памяти также составляет $4T$, однако каждый последующий квант выдается относительно предыдущего с задержкой всего T. В современных больших ЭВМ наиболее часто используют память, включающую в себя от 4 до 16 модулей. Но технология классического расслоения памяти практически не используется в персональных компьютерах из-за ее сложности и различных форматов команд.

Еще один способ ускорения обращения к памяти состоит в использовании широкого слова. При выборке широким слоем за одно обращение к ОП производится одновременная запись или считывание нескольких команд или слов данных из широкой ячейки. Затем это широкое слово заносится в регистр (при считывании из памяти), из которого отдельные команды и слова данных последовательно используются процессором без дополнительных обращений к ОП. Выборка широким слоем распространена в высокопроизводительных машинах, но в персональных компьютерах не получила широкого распространения, так как при записи-считывании широкого слова требуется многоразрядная магистраль для передачи данных. В персональном компьютере все передачи данных между процессором и ОП осуществляются по стандартной магистрали, ширина которой составляет 16, 32 или 64 бита (без учета контрольных разрядов).

Статическая и динамическая память. В *статических* ОЗУ записанная информация хранится, пока на интегральную схему подается питание. Запоминающий элемент, или ячейка статического ОЗУ, представляет собой триггер (рис. 6.6). Этот триггер строится на основе четырех или шести транзисторов. Схема триггера на четырех транзисторах проще, обладает меньшей стоимостью, но у нее больший ток утечки и она более чувствительна к воздействию внешних ис-

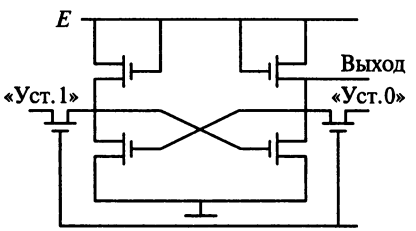


Рис. 6.6. Запоминающий элемент статической памяти

точников излучения. Два дополнительных транзистора служат не только для уменьшения перечисленных недостатков, но и повышают быстродействие такой схемы.

Динамическая память в качестве запоминающего элемента содержит конденсатор; кроме того, в ней используются три транзистора: для поддержания заряда конденсатора, записи и чтения. Но конденсатор не может удерживать заряд бесконечно долго, поэтому для предотвращения потерь информации производится периодическое восстановление заряда, или регенерация.

В качестве ОП в персональных компьютерах используются динамические ОЗУ, так как они дешевле и меньше по размерам, а меньшее быстродействие компенсируется наличием статической кэш-памяти.

Методы ускорения обменов с ОП. В персональных компьютерах долгое время использовалась динамическая память, работающая в режиме быстрого страничного обмена FPM DRAM. Собственно запоминающая часть такой памяти представляет собой множество ячеек, расположенных в виде прямоугольной матрицы. При чтении цикл обращения к ней начинается с активизации строки в запоминающей матрице, после чего выполняется активизация столбца адресуемой ячейки. Каждый прочитанный элемент данных проверяется на правильность и после этого передается процессору для обработки. Когда найдена нужная ячейка, столбец деактивируется и подготавливается к следующему циклу, что вызывает состояние ожидания. Следующий столбец в этой строке можно активировать, если предположить, что очередной квант данных расположен в соседней ячейке. В такой памяти цикл чтения четырех элементов, занимающих одну строку памяти, может выполняться за 15 тактов: на чтение первого элемента тратится шесть тактов, так как нужно активизировать строку и столбец, и по три такта на получение последующих трех элементов (для их чтения нужно активизировать лишь очередной столбец).

В основу технологии EDO DRAM (динамическая память с увеличенным временем доступности данных) положена та же память. Работа EDO DRAM начинается с активизации строки и столбца. Однако после нахождения элемента данных в памяти этого типа буфер данных остается включенным до обращения к следующему столбцу, т. е. устраняется состояние ожидания. Это позволяет сэкономить три такта при передаче пакета из четырех слов.

Самой быстрой представляется синхронная динамическая память (SDRAM). Основная особенность ее состоит в способности синхронизировать все операции с тактовыми сигналами процессора. Это уменьшает время обращения к столбцу памяти. Внутри памяти SDRAM находится счетчик, который увеличивается при адресации столбца, что инициирует новое обращение к памяти до завершения предыдущего. Все это позволяет затрачивать пять

тактов на чтение первого слова и по одному такту на чтение последующих слов, т.е. всего восемь тактов.

В последнее время появилась синхронная динамическая память с удвоенной скоростью передачи данных DDR SDRAM. В этой динамической памяти данные в пакетном режиме выдаются по обоим фронтам импульса синхронизации, за счет чего ее пропускная способность увеличивается вдвое. Этот тип динамической памяти в настоящее время является наиболее распространенным для персональных компьютеров.

Несмотря на большое число различных разработанных видов динамической памяти, отличающихся помимо принципов построения способами поиска и передачи информации во внешние цепи, скорость ее работы существенно отстает от скорости процессора. При одновременном увеличении тактовой частоты и ширины выборки возникают проблемы с электромагнитной совместимостью, труднее обеспечить одновременность получения всех информационных битов, сложнее организовать широкую шину между памятью и процессором. Поэтому обычно используют широкую шину (до 64 и даже 128 бит) при ограниченной частоте шины.

В памяти Rambus Direct RAM (DRDRAM) ширина выборки уменьшена до 16 бит, но частота увеличена до 800 МГц (или до 1600 МГц). Передача осуществляется пакетами: существует три вида пакетов — данных, строк и столбцов. Пакеты строк и столбцов служат для передачи команд управления, которые заменяют обычные сигналы RAS, CAS, WE и CS. Однако память Rambus используется редко из-за дороговизны и возникающих технологических сложностей. Существует множество других видов динамической памяти, предназначенных для будущих персональных компьютеров, например SLDRAM, ESDRAM, CDRAM. Эти виды памяти используют те же способы сокращения времени доступа, что и рассмотренные выше, но более высокая скорость передачи данных в них связана со значительным усложнением контроллера.

Кэш-память. Для достижения более высокого быстродействия между сравнительно медленной основной памятью, использующей динамические элементы, и относительно быстрым процессором размещают буферную память, получившую название кэша, или кэш-памяти. Ее использование позволяет избегать циклов ожидания в работе процессора, которые снижают производительность всей системы.

Кэш-память предназначена для кратковременного хранения информации и выдачи ее в процессор. Она не является программно доступной и служит только для повышения производительности, но не увеличивает общую емкость памяти и не влияет на программирование. Кэш-память бывает нескольких уровней. Кэш первого уровня *L*1 обычно интегрируется с процессором и слу-

жит для предварительной выборки команд и данных. Во многих процессорах кэш первого уровня разделен на два: кэш команд и кэш данных (см. рис. 5.10). Его объем в современных процессорах составляет от 4 до 64 Кбайт (и даже до 128 Кбайт), а время обращения около 10 нс. Когда говорят о кэш-памяти, то обычно подразумевают кэш второго уровня, или *L2*. Он служит для ускорения обращений процессора к программам и данным при их обработке. Этот кэш строится на базе очень быстрой и довольно дорогой статической памяти (SRAM) и хранит наиболее часто используемую процессором информацию. Физически кэш второго уровня выполняют в виде отдельной микросхемы или интегрированным с процессором в зависимости от возможностей технологии. Объем кэш-памяти второго уровня в процессорах разного назначения составляет от 256 Кбайт (в персональных компьютерах) до 8 Мбайт (в серверах), а время обращения 15...20 нс. В некоторых компьютерах предусматривается также кэш третьего уровня *L3*, однако его эффективность невелика.

Наличие кэш-памяти не должно вызывать затруднений при программировании и нарушать принятые принципы адресации. Чтобы скрыть от программиста наличие кэш-памяти, в ней используют ассоциативный принцип хранения и поиска информации.

В кэш-памяти располагается та же информация, что и в ОП. Любое обращение процессора к памяти вначале всегда направляется в кэш-память, и только при отсутствии в ней соответствующей информации продолжается ее поиск в ОП. Найденное слово передается в процессор для обработки, но при этом из ОП забирается целый блок информации, состоящий из нескольких слов, которые заносятся в кэш-память. При повторном обращении процессора к данным или командам из этого блока они уже будут находиться в кэш-памяти. Так как и команды, и данные обычно занимают последовательные ячейки памяти, высока вероятность, что следующее обращение в кэш-память будет успешным. При работе с кэш-памятью применяется ассоциативный принцип: старшие разряды адреса используются в качестве признака, а младшие — для выбора слова.

Существует три вида организации кэш-памяти: с прямым отображением, полностью ассоциативная и множественно ассоциативная.

В кэш-памяти *с прямым отображением* информационный блок из ОП записывается всегда в одно и то же фиксированное место. При записи блока в кэш-память используются младшие разряды его адреса ОП. Они также записываются в виде тега вместе с информацией. Для информации, хранящейся в ячейках ОП с одинаковыми младшими разрядами адреса, предназначен один блок кэш-памяти. Так, если ОП разделена на 16 384 блока, а объем кэш-памяти позволяет разместить только 128 блоков, то в строке 0

кэш-памяти могут находиться данные из 0, 128, 256,..., 16 256 блоков ОП, в строке 1 — данные из 1, 129, 257, ..., 16 257 блоков ОП и т.д. Такая кэш-память наиболее дешева, но и требует очень частой замены блоков; вероятность «попадания» сравнительно низкая. Кэш с прямым отображением используется в простых недорогих компьютерах.

Если информационный блок ОП может быть помещен на любое место кэш-памяти, то она называется *полностью ассоциативной.* Ее почти не используют из-за сложности схем управления.

В *множественно ассоциативной* кэш-памяти информационный блок ОП может располагаться на ограниченном числе мест. Обычно такое множество состоит из четырех каналов. В персональных компьютерах, как правило, используют именно четырехканальную множественно ассоциативную кэш-память. В ней сочетаются достоинства прямого и полностью ассоциативного отображения.

На рис. 6.7 приведена одна из возможных организаций множественно ассоциативной кэш-памяти. Такая память состоит из собственно блочного запоминающего устройства (БЗУ), в котором хранятся копии взятых в ОП информационных блоков, таблицы адресов (ТА) и управляющих схем. Рассмотрим операцию чтения. Поступающий из процессора адрес разбивается на три части: поле смещения (A_D), служащее для выбора нужного байта внутри блока, поле индекса($A_{инд}$), определяющее номер множества, и поле тега ($A_т$), т.е.

$$A = A_т, A_{инд}, A_D.$$

Обмен данными между кэш-памятью и ОП производится блоками. В таблице адресов (называемой также справочником) для каждого блока, расположенного в ЗУ по адресу $A_{инд}$, находятся адреса страниц, или тегов, $A_т$ тех блоков, которые были уже скопированы в кэш-память во время предыдущих операций чтения. При обращении процессора адрес запрашиваемой информации

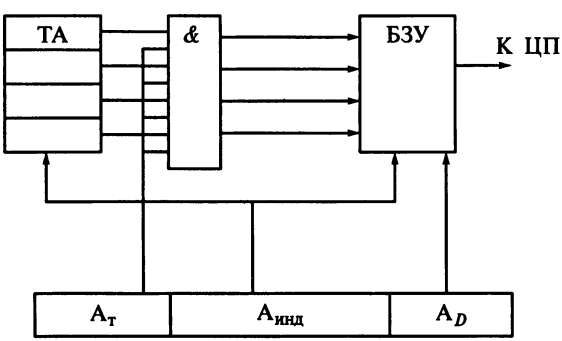


Рис. 6.7. Одна из возможных схем организации кэш-памяти

передается в регистр, где он разбивается на три части. Находящиеся в таблице адресов номера страниц сравниваются с номером страницы (т. е. старшими разрядами адреса) запрашиваемого блока. Если выявляется совпадение адресов страниц, то запрашиваемый блок находится в соответствующем разделе кэш-памяти.

Производится выборка этого блока по адресу $A = A_{инд}, A_D$. Если совпадения номеров страниц не произошло, то искомый блок отсутствует в ассоциативной памяти, и адрес нужного слова передается ОП. (Для ускорения работы памяти обычно адрес передается в кэш-память и ОЗУ одновременно, но если искомое слово находится в кэш-памяти, то там оно будет найдено быстрее, поэтому результат поиска в ОЗУ будет не нужен.)

Найденное слово из оперативной памяти передается в процессор для обработки, а весь блок переписывается в кэш-память. При копировании блока он направляется в раздел кэш-памяти, адрес которого определяется с помощью таблицы активности блоков (на рисунке она не показана). Эта таблица содержит предысторию обращений к каждому блоку и служит для вытеснения из кэш-памяти неиспользуемых блоков. Применяется одна из двух стратегий — случайная или LRU (least recently used — заменяется блок, который дольше других не использовался). В первом случае новый блок вытесняет из кэш-памяти любой блок, выбираемый случайным образом; эту стратегию значительно проще реализовать в аппаратуре. Во втором случае из кэш-памяти удаляется блок, к которому в течение последнего времени не было обращений. Эта стратегия требует фиксации обращений в таблице активности, что может значительно усложнить ее реализацию. Обычно реализация этой стратегии выполняется с помощью очереди, в которую заносятся ссылки на заполняемые строки кэш-памяти. При обращении к строке ссылка на нее перемещается в конец очереди. Таким образом, в начале очереди всегда будет находиться ссылка к строке, к которой обращение производилось раньше, чем к другим строкам.

Операции записи в память встречаются намного реже операций чтения; обычно они составляют менее 10 % общего числа операций, но пренебрегать ими нельзя. Существует две возможности:

сквозная запись (write through), при которой запись осуществляется одновременно и в кэш-память, и в ОП. В этом случае наличие кэш-памяти никак не повлияет на скорость работы машины при записи обработанной информации в память. Время, необходимое для записи слова в ОП, превышает длительность записи в кэш-память, а пока не завершена запись результата предыдущей операции, продолжать работу невозможно;

запись с обратным копированием (write back), при которой запись информации при обработке производится только в кэш-память. Это происходит быстро, но при этом в кэш-памяти и ОП

могут храниться различающиеся копии одних и тех же данных. Измененный блок из кэш-памяти передается в ОП только при его замещении. Для еще большего сокращения операций записи каждому блоку ставится в соответствие бит модификации, который отменяет запись блока в ОП, если он не изменялся.

Как правило, в персональных компьютерах сегодня используют сквозную запись, хотя она и не обеспечивает выигрыша во времени при записи. Это вызвано тем, что ее проще реализовать, чем запись с обратным копированием. Кроме того, при сквозной записи ОП и кэш-память всегда содержат одинаковые копии данных, что значительно упрощает процедуру ввода-вывода и организацию мультипроцессорных систем.

Внешняя память. Основная память машины строится на полупроводниковых элементах, которые сохраняют информацию только, если на них подается питание. При включении машины команды и данные загружаются в ОП из внешней памяти по командам BIOS. В качестве внешних запоминающих устройств наиболее распространены накопители на жестких магнитных и оптических дисках. Емкость памяти таких накопителей составляет несколько десятков и сотен гигабайт (современные НЖМД для персональных компьютеров имеют емкость свыше 100 Гбайт). Информация в этих накопителях записывается на поверхности диска подвижными головками записи-считывания в виде «отпечатков», образующих концентрические дорожки. Сами диски установлены на шпинделе, вращающемся с постоянной частотой, обычно составляющей 5400, 7200, 10 000 об/мин и выше. В настоящее время большинство накопителей для персональных компьютеров обладают частотой вращения дисков 7200 об/мин.

На каждой дорожке может располагаться несколько информационных блоков, как правило, фиксированного размера; эти блоки принято размещать в секторах. Обычно в секторе хранится 512 байт данных (в системе NTFS размер поля данных сектора соответствует 2048 байтам, в которых размещаются 1024 символа), а также служебная информация. Запись и считывание информационного блока производятся в тот момент, когда начало соответствующего сектора оказывается напротив головки. Задержка на поиск дорожки и ожидание, пока она окажется повернутой на угол, соответствующий началу сектора, называется *временем доступа*. Оно зависит от системы позиционирования головки, скорости вращения дисков, последовательности размещения блоков (чередование секторов, смещение дорожек), наличия или отсутствия дефектных дорожек и т. д. На это время влияет текучее положение головок относительно дисков, его среднее значение составляет от единиц до десятков миллисекунд.

После того как нужный блок будет найден, производится его чтение и передача в ОП. Скорость передачи составляет 1... 8 Мбайт/с.

Она зависит от размера блока, скорости вращения диска, плотности записи на дорожке, максимальной скорости передачи по шине и т.д. Время доступа превышает интервал между чтением или записью последовательных байтов блока на несколько порядков, т.е. основное время тратится не на передачу информации, а на поиск нужного сектора.

Накопители на магнитных дисках принято называть *устройствами памяти с прямым доступом*, так как при поиске сектора вначале осуществляется перемещение магнитной головки на соответствующую дорожку и только после этого производится поиск нужного сектора, для чего последовательно просматриваются все сектора на этой дорожке.

Для уменьшения затрат времени на поиск нужной информации используют традиционные методы буферизации и распараллеливания. Метод буферизации состоит в том, что в накопитель устанавливают буферное ЗУ, называемое *дисковым кэшем*. В этот кэш при чтении нужной информации из одного сектора диска помещают информацию из всех его секторов (или ряда последовательных секторов), находящихся на той же дорожке. На это требуется время, равное длительности одного оборота дисков. Если теперь понадобится информация из этих секторов, то она уже окажется прочитанной с диска в кэш и будет готова для передачи в ОП, т.е. не нужно будет тратить время на дополнительный оборот диска для ее поиска. Локальность программ и данных создает очень высокую вероятность обращений к диску за информацией, находящейся в смежных секторах. Иногда дисковый кэш организуют непосредственно в ОП, выделяя для этого специальную область. Такой способ уменьшения времени поиска не требует дополнительной аппаратуры, но сопряжен с затратами времени на передачу блоков в ОП и дополнительными расходами оперативной памяти.

При операциях записи информационный блок вначале переносится в дисковый кэш, а лишь затем (после выполнения соответствующих операций по поиску нужного сектора) на диск. Обычно его копия сохраняется в дисковом кэше до тех пор, пока не будет вытеснена другой. Запись блока в дисковый кэш может производиться только на свободное место, т.е. на место информационного блока, копия которого уже сохранена на диске.

6.3. Организация виртуальной памяти

Информация для обработки процессором может быть взята только из оперативной или кэш-памяти, поэтому при работе машины постоянно возникает необходимость обменов между ОП и внешней памятью в силу того, что объем ОП ограничен. Предусматривать такие обмены в настоящее время возлагается на средства ОС.

Страничное, сегментное и странично-сегментное распределение. Пользователю предоставляется адресное пространство, соответствующее всему объему внешней памяти. Это пространство получило название *виртуальной памяти.* Все виртуальное пространство разбивается на сегменты и страницы. Страницы имеют фиксированный размер и размещаются как в ОП, так и на дисках. Размер страницы обычно составляет 4 Кбайт и кратен емкости сектора на магнитном диске. Информация о текущем местонахождении страницы указывается в таблице (всегда находящейся в ОП) преобразования адресов. При простейшей страничной организации виртуальный адрес включает в себя номер страницы *VA* и смещение внутри нее A_D (рис. 6.8).

Номер страницы *VA* используется для обращения в таблицу преобразования адресов, каждая строка которой содержит указатель *X* наличия этой страницы в ОП и ее адрес. При наличии страницы в ОП, т.е. когда бит указателя установлен в единицу, происходит обращение к ячейке по адресу $A = A_{стр}, A_D$.

Если бит указателя сброшен в положение «0», то номер этой страницы виртуального пространства передается в блок замены страниц, который выбирает в ОП удаляемую страницу и заносит на ее место нужную страницу из внешней памяти; после этого он модифицирует содержимое таблицы преобразования адресов, устанавливая «1» в разряд указателя данной страницы. Поскольку размер страницы фиксирован, а размеры программ могут быть разными, то страничная организация памяти неудобна. Более приемлемой оказывается сегментно-страничная организация памяти,

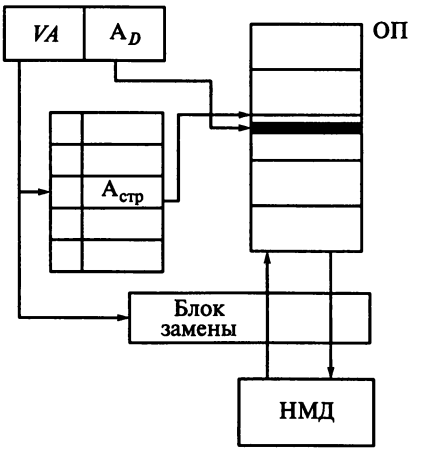


Рис. 6.8. Организация виртуальной памяти

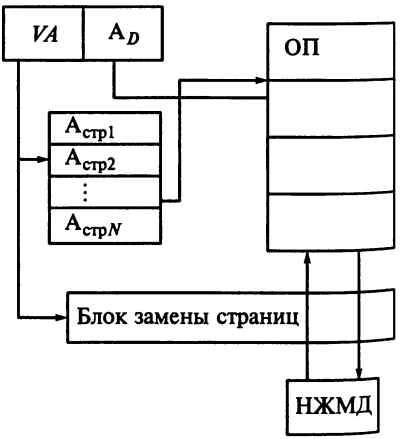


Рис. 6.9. Преобразование адреса при сегментно-страничной организации памяти

при которой каждый сегмент, т.е. выделенный участок адресного пространства для какой-либо программы, разбивается на определенное число страниц фиксированного размера. Виртуальный адрес состоит из номера сегмента, номера страницы и смещения. Преобразование виртуального адреса в физический производится с помощью таблицы сегментов и таблицы страниц (рис. 6.9).

Для ускорения процесса преобразования виртуального в физический адрес могут использоваться специальные аппаратные средства на базе ассоциативной памяти. Номер страницы *VA* виртуального адреса передается в ассоциативную память в качестве поискового признака; вторым полем этой таблицы служит адрес страницы в ОП. Если в таблице отсутствует нужный виртуальный адрес, то преобразование осуществляется, как описано выше. Этот механизм называется *динамическим преобразованием адресов.*

Стратегия замены страниц оказывает сильное влияние на производительность всего компьютера, особенно, если объем оперативной памяти невелик. Такая стратегия определяет, как выбирать подлежащую замене страницу в ОП на новую страницу из дисковой памяти. Эти стратегии основаны на анализе моментов записи страницы в ОП (FIFO) или анализе моментов времени, когда к странице осуществлялось последнее обращение (LRU).

Свопинг. Объем оперативной памяти ограничен, но чтобы выполнить задачу, необходимо загрузить ее в ОП. Обычно процессор, работающий в режиме мультипрограммирования (например, в сервере), выполняет несколько задач: одна задача находится на стадии выполнения в процессоре, а для других осуществляется ввод-вывод или они находятся в стадии ожидания. Из-за ограниченности объема ОП повышать степень мультипрограммирования оказывается невозможно.

Преодолеть эту трудность помогает *свопинг*, т.е. способ организации вычислительного процесса, при котором задачи, находящиеся в состоянии ожидания, переносятся на жесткий диск, освобождая часть ОП. При этом планировщик операционной системы следит за изменением условий, складывающихся для этих задач, а при благоприятных условиях (т.е. когда можно занять ресурсы процессора, для данной задачи завершен ввод-вывод, свободны необходимые устройства для выполнения задачи и т.д.) он возвращает задачу из области свопинга на диске в ОП. Таким образом, свопинг можно рассматривать как своеобразный способ организации виртуальной памяти.

6.4. Защита памяти

Этот механизм принято называть защитой памяти, на самом деле осуществляется защита хранящейся в ней информации. Обыч-

но в процессорах предусматривают специальные средства для защиты памяти от несанкционированного доступа со стороны других программ. В памяти компьютера всегда находится несколько программ, даже если он выполняет только одну. Из-за ошибок или по каким-либо иным причинам программа может вторгаться в «чужую» область памяти, искажая находящуюся там информацию. Чтобы этого не происходило, в компьютере предусматривают систему привилегий, которая регулирует доступ к той или иной области памяти в зависимости от уровня ее защищенности и привилегированности запроса к ней.

Защиту отдельных ячеек памяти можно осуществить, выделяя в каждой ячейке специальный разряд защиты. Наличие единицы в этом разряде приводит к блокированию записи в эту ячейку и сообщении об ошибке, тем самым обеспечивается защита ячейки от попыток записи. Но этот механизм неудобен и в современных компьютерах практически не используется.

Вторым механизмом защиты служат так называемые кольца защиты. Так, в микропроцессоре Pentium (как и в ранее выпускавшихся микропроцессорах 286, 386 и 486, а также в ЕС ЭВМ и ряде других машин) предусмотрены четыре кольца, или уровня привилегий, имеющие номера от 0 до 3. Чем меньше номер, тем выше уровень защищенности.

Наивысшим уровнем защиты обладает ядро ОС; ему присвоен уровень 0. В это ядро входят команды, обеспечивающие инициализацию системы, управление доступом в память и ряд других важнейших функций. Уровень 1 имеют утилиты ОС, а к уровню 2 обычно относят всевозможные служебные программы, драйверы, систему управления базами данных и т. п. Наименее защищены прикладные программы пользователя; им присваивается уровень с номером 3.

Для доступа к программам и данным, хранящимся в ОП, используются следующие правила:

данные из сегмента ОП с некоторым уровнем защиты могут быть выбраны программой с таким же или более высоким уровнем привилегий;

программная процедура может быть вызвана программой с таким же или более низким уровнем привилегий.

Уровни защиты определяются двумя битами (указывающими уровень защищенности). При обращении в ОП должно быть произведено сравнение разрешенного уровня запроса (указывается в двух битах специального описателя) с фактическим (приведенным в специальном регистре-селекторе). Если уровень запроса, указанный в селекторе, ниже разрешенного уровня, то данные ячейки ОП оказываются закрытыми для доступа к ним. Доступ к менее привилегированной процедуре может осуществляться только через специальный шлюз вызова.

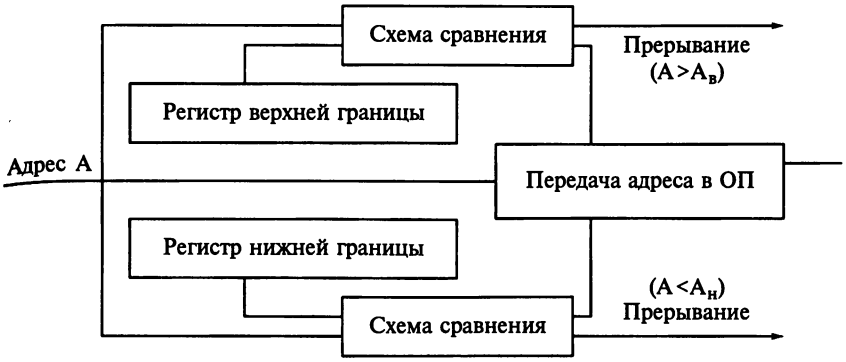


Рис. 6.10. Защита памяти методом граничных регистров

Самый распространенный метод защиты памяти — *метод граничных регистров.*

В процессоре предусматривают два специальных регистра, содержащих верхнюю и нижнюю границы области памяти, к которой может обращаться текущая программа (рис. 6.10). Занесение границ в эти регистры производится ОС в привилегированном режиме при загрузке программы.

При запросе информации из ОП, осуществляемом в пользовательском режиме, адрес сравнивается с установленными границами и, если он находится между верхней и нижней границами, передается в память. Если же происходит нарушение защиты, т.е. адрес выходит за установленные рамки, то доступ в память блокируется и формируется сигнал нарушения защиты.

Для организации защиты несмежных областей памяти в машинах ЕС ЭВМ использовали ключи защиты — каждому блоку памяти присваивался некоторый код, или ключ защиты. Каждая программа получала свой код — код защиты программы. Доступ к любому блоку памяти возможен только при совпадении значений ключа и кода защиты или при равенстве ключа «0». Нулевое значение ключа определяет возможность доступа ко всему адресному пространству памяти со стороны ОС. Код (или ключ) защиты программы находится в специальном регистре, хранящем слово состояния программы. При обращении к ОП производится сравнение ключей защиты памяти и программы: при совпадении ключей доступ к памяти разрешается, а при несовпадении — формируется сигнал нарушения защиты.

Контрольные вопросы

1. Как принято определять объем оперативной памяти? Почему при указании объема памяти компьютера не учитывают кэш-память?

2. Почему в современных персональных компьютерах используют различные виды памяти: кэш-память, оперативную, внешнюю?

3. Что такое адресное пространство? Как адресуется различная память в современных персональных компьютерах?

4. С какой целью в адресном пространстве предусматривают адреса постоянной памяти? Что хранится в ПЗУ?

5. Как можно повысить скорость обращения к ОП? Что такое расслоение памяти?

6. Какие существуют принципиальные ограничения на расслоение памяти и к чему они приводят?

7. Каким образом осуществляется контроль правильности информации, находящейся в ОП, и в какой момент он осуществляется?

8. Как организована ОП и какой объем информации можно получить за одно обращение к ней?

9. На каких элементах строится ОП в персональных компьютерах и чем это обусловлено? С какой целью выполняется регенерация памяти?

10. Какие существуют виды кэш-памяти? Почему в настоящее время чаще всего используют множественно-ассоциативную кэш-память?

11. Как и в какой момент нужно производить запись информации в ОП из кэш-памяти? Что такое сквозная запись и запись с обратным копированием?

12. Какой выигрыш можно получить при использовании записи с обратным копированием? Для чего она неудобна?

13. Приведите примеры характеристик внешней памяти на жестких дисках. Каким образом на них представлена информация?

14. Что такое виртуальная организация памяти? Каким образом осуществляется использование информации из виртуального пространства процессором?

15. Какими средствами и как можно ускорить преобразование виртуального адреса в физический?

16. Почему в персональных компьютерах используют кэш-память нескольких уровней? Какими объемами характеризуется такая кэш-память *L*1 и *L*2?

17. Какие функции реализуют кэш-память *L*1 и кэш-память *L*2? Какой объем кэш-памяти у современного персонального компьютера?

18. Что такое страничная организация памяти? Как она осуществляется? Чем она неудобна?

19. В чем заключаются преимущества сегментно-страничной организации памяти?

20. Что такое флэш-память? Где она используется и для каких целей?

21. Что такое защита памяти? Как она выполняется в машинах с процессором Pentium?

22. Какие методы защиты памяти используют в современных персональных компьютерах и в чем их сущность?

23. Каким образом обеспечивается защита информации в памяти от несанкционированного доступа?

24. Что такое «свопинг» и как он осуществляется?

ГЛАВА 7

ИНТЕРФЕЙСЫ

7.1. Понятие интерфейса и его характеристики

В первых компьютерах АЛУ соединялось с памятью и различного рода устройствами ввода-вывода так, как хотели этого разработчики машины. Однако начиная с середины 1960-х гг. ситуация меняется коренным образом: подключение периферийных устройств осуществляется по общим и единым правилам. Это открыло пути к массовому производству периферийных устройств, а саму машину превратило в *систему с переменным составом оборудования*.

Совокупность таких правил, а также аппаратные, программные и конструктивные средства для их реализации принято называть *аппаратным интерфейсом*, или *интерфейсом ввода-вывода*. Затем в мини- и микроЭВМ идеи стандартизации обменов информацией получили дальнейшее развитие. Все обмены между процессором, памятью и периферийными устройствами происходили по единой системе соединений, получившей название *шины*. Впервые это название было предложено фирмой DEC при создании компьютера PDP-11.

Принято различать физический и логический интерфейсы. *Физический* интерфейс представляет собой совокупность механических и электрических аппаратных средств, а также физических сред передачи сигналов, а *логический* — совокупность правил передачи кодированной информации между устройствами, узлами или элементами системы, т.е. протоколы взаимодействия, или алгоритмы формирования сигналов обмена.

Интерфейсы принято характеризовать:

видом связи, т.е. способностью вести дуплексную (сообщения могут одновременно передаваться в двух направлениях), полудуплексную (линии связи используются для двухсторонней передачи, но сообщения передаются в разные моменты времени) или симплексную (передача возможна только в одном направлении) передачу;

пропускной способностью, т.е. количеством информации, которое можно передать по линиям интерфейса в единицу времени;

максимальной частотой передачи сигналов данных;

динамическими параметрами — временем передачи отдельного слова, блока, пакета;

максимально допустимым расстоянием или суммарной длиной линий, соединяющих все устройства интерфейса;

общим числом линий в интерфейсе;

допустимым числом подключаемых устройств;

задержками при организации передачи, вызванными необходимостью выполнять подготовительные и завершающие действия при установлении связи между устройствами.

Важной составной частью физического интерфейса являются линии связи для передачи сигналов. *Линия связи* — это физическая среда, по которой осуществляется передача сигналов от одного или нескольких источников к одному или нескольким получателям информации. Такими линиями связи для передачи сигналов могут быть провода, витая пара, медные полосковые соединительные линии на печатной плате, оптоволокно, радио или инфракрасные каналы.

Группу линий связи, обеспечивающих передачу команд и данных между устройствами компьютера, принято называть *шиной*. Существует два типа шин: параллельная и последовательная. По параллельной шине, состоящей из нескольких линий связи, передача сообщения происходит квантами, содержащими по m бит. Наиболее распространены интерфейсы, в которых одновременно могут передаваться 8, 16, 32 и 64 бита. В последовательном интерфейсе передача сообщения производится всего по одной линии, хотя общее число линий может быть значительно больше. По ним передаются сигналы синхронизации и управления. Последовательные интерфейсы характеризуются меньшими скоростями передачи информации, чем параллельные, но обычно позволяют передавать сообщения на значительно большие расстояния.

Компонентами любого интерфейса принято считать:

шину (физические линии) и разъемы (слоты);

форму и параметры сигналов;

электронные схемы (контроллеры и адаптеры);

алгоритмы управления.

В первых персональных компьютерах передача команд и данных осуществлялась по единой шине. Однако вскоре стало ясно, что наличие всего одной шины, к которой подключаются и быстрые, и медленные устройства, ограничивает его возможности. Тогда-то и стали применяться в персональных компьютерах несколько различных шин, к которым все устройства подключаются с помощью специальных контроллеров. Передача информации через эти интерфейсы осуществляется с помощью сигналов, имеющих различные параметры, и посредством различных алгоритмов.

Наибольшее развитие система интерфейсов получила в персональных компьютерах. Поскольку в их работе принимают участие

как внутренние, так и внешние устройства, и нужно обеспечить передачу информации между любыми из них, то интерфейсы персональных компьютеров принято подразделять на внутренние, локальные и внешние.

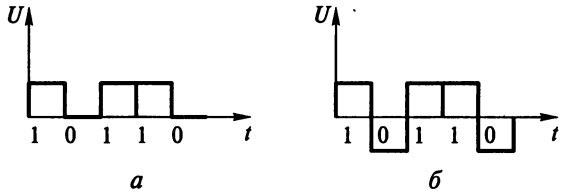
К *внутренним* интерфейсам принято относить системную шину (это интерфейс между ЦП и главным контроллером), шину системной памяти (интерфейс между главным концентратором и ОП), шину графического процессора AGP (интерфейс между главным контроллером и графическим процессором), интерфейс для подключения контроллера ввода-вывода. В число внутренних включают также универсальный интерфейс PCI (Peripheral Component Interconnect), служащий для подключения контроллеров разнообразных ПУ.

Локальными называют интерфейсы для подключения накопителей на жестких дисках (IDE/ATA и SCSI), а также звуковых и модемных кодеков.

К *внешним* интерфейсам относят последовательный порт COM (RS-232), параллельный порт LPT (IEEE 1284), универсальный последовательный интерфейс USB, интерфейс FireWire (IEEE 1394) и ряд других, служащих для подключения ПУ. Эти интерфейсы часто называют *интерфейсами ввода-вывода*. Они предназначены для подключения внешних устройств, например клавиатуры, мыши или дисплея к центральной части компьютера.

Состав линий системной шины. Среди внутренних интерфейсов «узким местом» всегда являлась системная шина, поскольку практически все устройства конкурируют за возможность передачи по ней своих данных. *Системная шина* (иногда ее называют магистралью) — это среда передачи данных между ОП и устройствами ввода-вывода. К ней параллельно могут подключаться несколько устройств. В персональных компьютерах системная шина выполняется в виде печатных проводников на общей системной плате, и ее иногда называют материнской или объединительной. Она может содержать десятки и сотни линий. Физически она состоит из трех подшин, иногда называемых просто шинами: информационной, адресной и управления. Информационную и адресную шины часто выполняют совмещенными, т.е. по одним и тем же проводам (или линиям) в разные моменты времени передаются и адреса, и данные. Особенно часто совмещение шин реализуют, когда шина строится в виде параллельных печатных проводников на плате и предназначена не только для обращения к ПУ, но и к ОП. Это вызвано сложностью и сравнительно высокой ценой размещения на плате большого числа информационных и адресных линий, необходимых при обращениях к памяти.

Передача данных по проводным линиям связи. По линиям связи современных интерфейсов преимущественно передаются низкочастотные дискретные уни- и биполярные сигналы (рис. 7.1).



а *б*

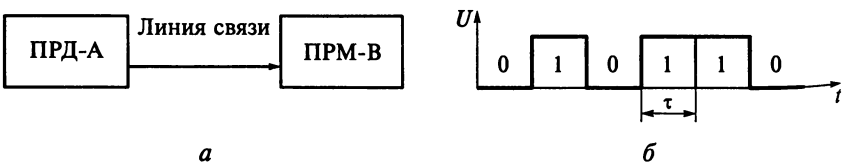
Рис. 7.1. Временно́е представление сигналов:
а — униполярных; *б* — биполярных

Одиночный импульс можно характеризовать длительностью τ, амплитудой U, а также шириной спектра F_c и динамическим диапазоном сигнала D_c. Для повышения скорости обмена необходимо уменьшать длительность импульса τ, так как скорость передачи данных в линии связи определяется выражением $V_п = 1/\tau$. Однако вероятность определения приемником наличия импульса в данном дискретном временно́м интервале является функцией энергии принятого импульса, которая тем больше, чем больше площадь отображения передаваемого импульса $E = U\tau$. Это требует увеличения его длительности τ, так как увеличение амплитуды сигнала U ведет к росту энергопотребления и возрастанию помех.

Используются два метода передачи данных через интерфейс: синхронный (изохронный) и асинхронный.

При *синхронном* методе (рис. 7.2, *а*) передающее устройство (ПРД-А) устанавливает одно из двух возможных состояний сигнала, например «0» — представляется паузой, а «1» — импульсом положительной полярности, и удерживает его в течение определенного, заранее оговоренного для передатчика и приемника (ПРМ-В) времени τ, только по истечении которого состояние сигнала на передающей стороне может быть изменено (рис. 7.2, *б*).

При использовании синхронного метода время передачи сигнала складывается из времени распространения сигнала по линии связи $t_{л.с}$ и времени распознавания и фиксации в регистре приемного устройства $t_р$, зависящих от параметров сигнала, линии связи и характеристик приемного устройства. Следовательно, интервал τ должен быть выбран из условия $\tau \geq T_{max} = t_{л.с} + t_р$, где T_{max} —



а *б*

Рис. 7.2. Синхронный метод передачи сигналов:
а — синхронный канал; *б* — диаграмма работы

максимальное время передачи сигналов (с учетом возможных наихудших условий).

При асинхронной передаче (рис. 7.3, *а*) ПРД-А устанавливает соответствующее передаваемому символу состояние сигнала на линии, а приемное устройство ПРМ-В после приема состояния сигнала информирует об этом ПРД-А изменением сигнала на линии (В → А). Это позволяет ПРД-А, не дожидаясь истечения установленного времени τ (рис. 7.3, *б*), перейти к изменению сигнала в соответствии с очередным символом сообщения (таким образом система передачи адаптируется к качеству канала связи). Поэтому асинхронный метод, несмотря на требования передачи сигналов в обоих направлениях, является более быстродействующим, но приводит к усложнению устройств обмена.

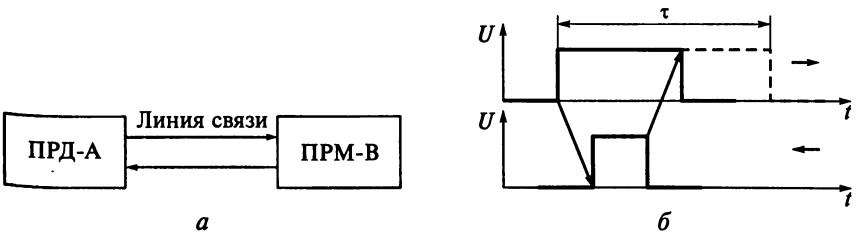
Еще одной проблемой является передача сигналов параллельного кода по нескольким линиям интерфейса. Передаваемые по разным линиям сигналы неизбежно, вследствие разброса параметров линий, поступают в приемное устройство в разное время, неодинаковым будет и время фиксации сигналов в регистрах приемного устройства, при этом максимальный разброс времени передачи

$$\Delta t = \max\{|t_i - t_j|\},$$

где t_i, t_j — время передачи по линиям i и j соответственно ($i, j = \overline{j, n}$; $i \neq j$).

Уменьшение τ при увеличении тактовой частоты может привести к тому, что линейно-тактовая синхронизация ПРД и ПРМ нарушается и передача сообщений будет существенно затруднена или нарушена вообще. Это одна из самых главных проблем параллельных интерфейсов.

Синхронный и асинхронный способы передачи выполняются с использованием методов стробирования (синхронная передача) и квитирования (асинхронная передача).



а *б*

Рис. 7.3. Асинхронный метод передачи сигналов:
а — асинхронный канал; *б* — диаграмма работы

Рассмотрим метод стробирования на примере передачи адреса от активного устройства к пассивному по магистрали «общая шина», состоящей из 16 параллельных линий связи A00...A15 (рис. 7.4).

В момент времени t_0 активное устройство начинает выдачу на линии адресной шины A00...A15 напряжений низкого уровня (т. е. кода 111...1), поскольку в магистрали «общая шина» напряжение низкого уровня (порядка 0,5 В) соответствует логической «1». С учетом разброса параметров передающих усилителей и других помех сигналы на всех линиях адресной шины примут истинные значения логической «1» только к моменту времени t_1.

В момент времени t_2 (не менее чем через 75 нс после момента t_1 для магистрали ТТЛ-типа длиной 15 м, согласованной на концах; $\Delta t = t_1 - t_0 \leq 75$ нс) активное устройство формирует напряжение низкого уровня на линии управления MSIN, подтверждая по отрицательному фронту сигнала передачу адреса. В этот момент времени существует гарантия того, что переходные процессы на всех линиях адресной шины закончились как минимум на 75 нс раньше. Эти 75 нс могут использоваться для дешифрации адреса всеми пассивными устройствами. С учетом этого отрицательный фронт сигнала MSIN может использоваться в качестве строба (синхроимпульса) для опроса дешифраторов адресов.

При получении ответного сигнала от единственного пассивного устройства, которое восприняло выданный адрес как «свой», активное устройство снимает сигнал MSIN, а затем в промежутке времени $t_3...t_4$ освобождает адресную шину. В пассивном состоянии на всех линиях адресной шины устанавливается напряжение высокого уровня (приблизительно 3,5 В).

Адресация и идентификация. Одним из характерных способов группового обмена по общей шине является централизованный, когда по общей шине устройство A_0 обменивается данными с одним из устройств $B_i (i = \overline{1,n})$, подключенным к магистрали. В этом

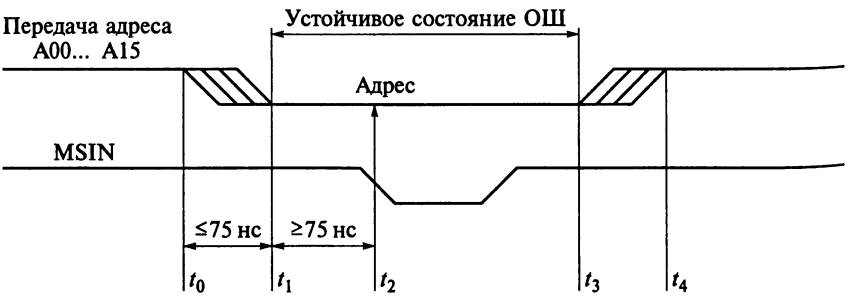


Рис. 7.4. Передача адреса по параллельной магистрали со стробированием

случае возникает необходимость в решении задач адресации и идентификации.

Адресация — это выбор центральным устройством A_0 одного из устройств B_i для связи. *Идентификация* состоит в определении центральным устройством A_0, какое из устройств B_i запрашивает связь. Решение этих двух задач требует передачи по линиям управления интерфейса дополнительной информации, количество которой зависит от организации и структуры интерфейса.

Возможен интерфейс радиального типа с индивидуальными линиями, в котором центральное устройство соединено с периферийными по отдельным линиям. При использовании такой системы существенно упрощается решение задач адресации и идентификации, но при этом увеличивается количество оборудования, следовательно, повышаются стоимость, энергопотребление и снижается надежность.

При использовании схем с коллективными линиями значительно уменьшается их число, но существенно усложняются процессы адресации и идентификации. В этом варианте ПУ B_i принимают все предаваемые центральным устройством A_0 сообщения, но селектируют их в зависимости от собственного адреса. Идентификация активных ПУ центральным устройством реализуется путем селекции соответствующего идентификационного номера. В этом случае шина управления служит для определения момента, когда возможна передача данных, нужно передать адрес, выполнить процедуру арбитража (т. е. определить, какое из подключенных к шине устройств имеет право выставлять свои данные на шину) и т. п.

К системной шине всегда подключается более двух устройств, а одновременно передача данных может производиться только между двумя из них. Это означает, что в один и тот же момент возможно появление нескольких запросов, которые могут быть удовлетворены только последовательно один за другим, поэтому каждое из устройств должно получить в свое распоряжение шину и только после этого передавать по ней данные. Следовательно, любая шина должна обладать средствами арбитража, позволяющими выстраивать очередь запросов, разрешая одновременную передачу данных только между двумя устройствами.

Арбитраж можно реализовать разными способами: чаще всего его организуют в виде последовательного распределенного или параллельного арбитража.

При *последовательном* распределенном арбитраже все устройства подключаются последовательно к линии, по которой передается арбитражный сигнал $АРБ_N$ (рис. 7.5). Этот сигнал подается на выход устройства, которому присвоен наивысший приоритет и подключается к точке с потенциалом земли, а выходной сигнал $АРБ_0$ подается на вход следующего устройства с более низким

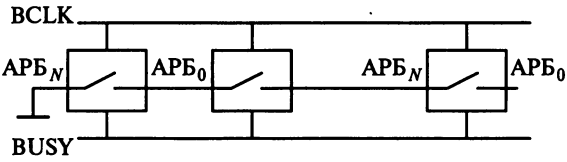


Рис. 7.5. Схема последовательного распределенного арбитража

приоритетом. Таким образом, образуется цепочка последовательно подключаемых устройств. Сигнал АРБ$_N$ подается в эту цепочку и достигает устройства, которое должно стать задатчиком, т.е. устройством, занимающим интерфейс для передачи или приема данных. На все устройства подается синхронизирующий сигнал BCLK; устройство имеет право выставлять запрос на занятие шины, т.е. размыкать ключ в цепи арбитражного сигнала по положительному фронту сигнала BCLK.

По отрицательному фронту этого сигнала устройство формирует сигнал на линии BUSY, захватывая магистраль, т.е. теперь магистраль отдается в распоряжение исключительно этого устройства. Таким образом, чтобы устройство стало задатчиком ЗДТ, должны быть выполнены следующие условия:

отсутствие сигнала АРБ$_0$ (т.е. устройство запрашивает шину);

присутствие сигнала АРБ$_N$ (т.е. ни одно из более приоритетных устройств не запрашивает шину);

отсутствие сигнала BUSY (т.е. шина свободна).

Для организации *параллельного* арбитража используется специальная схема приоритетного шифратора-дешифратора (рис. 7.6, *а*). На входы шифратора *CD* подаются запросы от множества устройств на право занять шину BREG, а на выходах дешифратора *DC* может присутствовать только один сигнал BPRN, направляемый к одному из подключенных к нему устройств и разрешающий использовать шину в данный момент для передачи данных (рис. 7.6, *б*). Устройство, получившее этот сигнал, захватывает магистраль (т.е. выставляет сигнал BUSY), становясь задатчиком. Этот сигнал удерживается до завершения работы устройства, предотвращая занятие шины другими устройствами. Схема параллельного арбитража ограничивает число устройств количеством входных и выходных сигналов. Однако она обладает высоким быстродействием и обычно служит для арбитража контроллеров прямого доступа в память.

Помимо параллельного арбитража известна схема циклического арбитража. Ее работа подобна работе схемы параллельного арбитража за исключением того, что после завершения цикла работы, т.е. когда какому-нибудь устройству разрешено передавать данные по магистрали, ему присваивается самый низкий приоритет, а приоритет всех остальных устройств увеличивается. Число уст-

ройств при параллельном и циклическом арбитраже ограничено числом входов и выходов приоритетного шифратора.

Непосредственно к шине подключается не устройство, а его контроллер (или адаптер); само же устройство подключают к этому контроллеру через точку доступа, которую называют физическим портом. *Физический порт* — это место подключения устройства, т. е. практически это разъем для подсоединения устройства к линиям интерфейса. Порт, или малый интерфейс, позволяет подключать устройство к контроллеру стандартным образом. В персональных компьютерах ПУ подключают к параллельному или последовательному интерфейсу.

Как правило, малые интерфейсы строят по радиальному принципу, т. е. устройство подключается по отдельным принадлежащим только ему проводам, а поэтому необходимость в адресации отпадает (к каждому интерфейсу может подключаться только одно устройство).

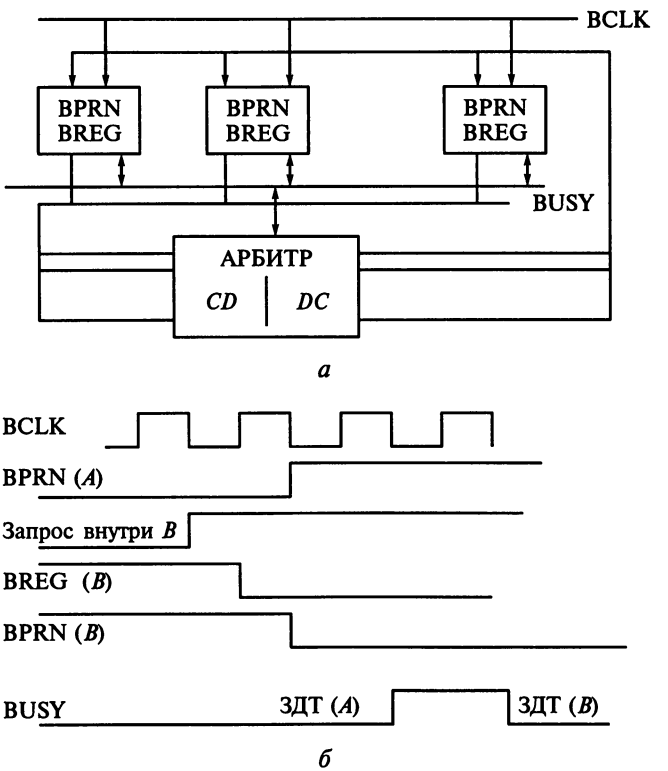


Рис. 7.6. Параллельный арбитраж:
а — схема работы; *б* — временна́я диаграмма

7.2. Подключение устройств

В зависимости от степени участия ЦП в обмене данными между памятью и ПУ может быть реализовано три способа управления обменом: асинхронный, синхронный и режим прямого доступа в память (ПДП). При *асинхронном* режиме производится последовательный опрос, или сканирование, всех ПУ процессором с целью узнать готовность его к обмену. Если ПУ готово к обмену данными, то ЦП организует процедуру программного обмена, т. е. загружает соответствующую программу-драйвер, производит пересылку данных между ПУ и ОП, вносит необходимые изменения в программу и выгружает ее в память. Если же ПУ не готово к обмену данными, то ЦП продолжает опрос других ПУ. Такой режим сканирования существенно упрощает организацию обмена, но ведет к значительной занятости ЦП.

При *синхронном* обмене ЦП также выполняет все функции по управлению обменом, но, в отличие от режима сканирования, реализация этих функций начинается с приходом сигнала прерывания от ПУ. Центральный процессор выполняет какую-либо программу и регулярно проверяет специальный регистр на наличие сигнала прерывания. Как только ПУ сообщает о своей готовности к обмену сигналом прерывания, ЦП производит переключение программ, переходит на обработку прерывания, осуществляет управление обменом и возвращается к выполнению прерванной программы. Такой обмен называют обменом по прерыванию.

Для ввода-вывода данных от быстродействующих устройств (например, внешней памяти) используют режим *прямого доступа в память*. Для реализации прямого доступа необходим специальный контроллер ПДП, в который перед началом обмена ЦП передает адрес ОП и количество передаваемых данных. После этого ЦП отключается от контроллера ПДП, передав ему процесс управления обменом, а сам продолжает выполнение своей программы. После окончания обмена контроллер ПДП сообщает об этом ЦП, для чего вновь прерывает его работу.

Возможно несколько способов организации интерфейсов:

индивидуальные интерфейсы для каждого из ПУ;

общий интерфейс;

несколько интерфейсов на разных уровнях иерархии.

Индивидуальные интерфейсы для каждого ПУ ведут к значительному увеличению загрузки ЦП и наличию очень большого числа линий. Они практически не используются не только в персональных компьютерах, но и вообще в вычислительной технике.

Общий интерфейс предполагает наличие единой шины, к которой подключаются все устройства, при этом подключение ПУ осуществляется через контроллеры. Эта единая шина служит также для соединения между собой ЦП и ОП (рис. 7.7, *а*). На этом

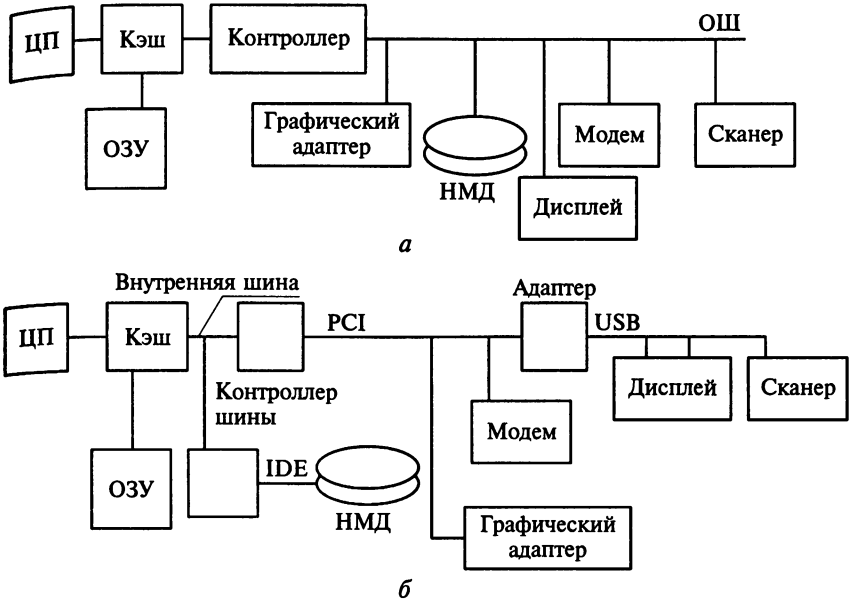


Рис. 7.7. Персональный компьютер:
а — с единой шиной; *б* — с локальной шиной

рисунке условно не показаны контроллеры ПУ (подразумевается, что каждое устройство имеет собственный контроллер, обеспечивающий подключение к шине).

Выводы ЦП должны подключаться непосредственно к контактам шины, к которым подключены и все другие устройства. Эта структура устраняет недостатки первой, но в каждый момент через общую единую шину могут обмениваться только два устройства, т.е. она становится общим ресурсом, что ведет к снижению производительности компьютера. Во время операции передачи данных между ПУ и ОП шина оказывается занятой, а следовательно, связь процессора с памятью блокированной.

В настоящее время самое широкое использование получил оптимизированный по пропускной способности способ иерархической организации интерфейсов. На схеме такого способа с локальными шинами (рис. 7.7, *б*) выделены интерфейсы IDE, PCI и USB.

При современной технологии все интерфейсы организуются посредством БИС. Так, системная шина соединяет ЦП с главным концентратором, который, в свою очередь, соединяется с памятью, графическим процессором и концентратором ввода-вывода. Концентратор ввода-вывода имеет интерфейсы PCI, AGP, Serial ATA, Ultra ATA, USB и др.

7.3. Внутренние интерфейсы

Шина PCI. Шина PCI (Peripheral Component Interconnect — межсоединение периферийных компонентов) является внутренним интерфейсом, т.е. к ней подключаются контроллеры ПУ, находящиеся внутри компьютера. Внутренний интерфейс всегда считался «узким местом» системы, так как практически все устройства компьютера конкурируют за возможность передачи по ней своих данных в память.

Вначале в IBM PC использовалась общая шина стандарта ISA (Industry Standart Architecture) с тактовой частотой 8 МГц, способная передавать от 8 до 16 Мбайт/с (ISA AT). Но вскоре ее пропускная способность оказалась недостаточной, и был предложен стандарт EISA (Extended ISA), обеспечивающий скорость передачи данных до 33 Мбайт/с. Тем не менее такой пропускной способности тоже не «угнаться» за скоростью работы процессора и памяти.

Для решения этой проблемы был разработан новый подход, состоящий в том, что шина для подключения ПУ не была напрямую связана с внутренней системной шиной. Это позволило обеспечить независимость интерфейса от конкретного типа ЦП. Однако новый интерфейс был несовместим ни с одним из предшествующих.

Этот интерфейс, или локальная шина PCI, служит для ускорения обращений процессора к ПУ. К ней подключаются устройства, скорость работы которых может превышать быстродействие шины ISA. Шина PCI связана с внутренней системной шиной процессора посредством специального набора микросхем, выполняющего функции контроллера шины. Этот набор микросхем (дословный перевод английского термина «chipset») содержит память достаточно большого объема, позволяющую буферизовать передаваемые данные, изменять их формат, задерживать передачу и т.п.

Выпускается несколько различных интегральных схем, выполняющих функции такого набора. Наиболее известные интегральные схемы реализуют 64-разрядный канал доступа к памяти, обеспечивают параллельную работу шины PCI, а также содержат средства сопряжения с устройствами ввода-вывода. Эти средства обеспечивают работу интерфейса USB, программируемого ввода-вывода и управления шиной дискового накопителя. Современные наборы позволяют адресовать до 4 Гбайт ОП, применять память с коррекцией ошибок и управлять двухпроцессорной конфигурацией. Подобные микросхемы регулярно обновляются; на них возлагается все большее число функций.

Шина PCI обладает довольно высокой пропускной способностью и обеспечивает поддержку режима P&P (Plug-and-Play — «включай и работай»). Первоначально тактовая частота шины составляла 33, а позднее 66 МГц. В настоящее время шина PCI-X

работает с тактовой частотой 133 МГц, обеспечивая пропускную способность до 1064 Мбайт/с при передаче 64-разрядных данных слов.

Конструктивно шина PCI выглядит в виде нескольких разъемов на системной плате, между которыми находятся печатные проводники. Эти разъемы сгруппированы в сегменты; число разъемов в сегменте не превышает четырех. Общее число разъемов также невелико (обычно не более 5...7), так как увеличение их числа привело бы к удлинению линий и, следовательно, снижению тактовой частоты. В качестве ПУ могут использоваться специальные «мосты» для организации связи этой шины с другими видами шин.

Для повышения пропускной способности шины применяются несколько способов.

Во-первых, осуществляется блочная передача данных, хранящихся в последовательных ячейках. Для этого в БУ шиной PCI предусматривается таймер для нескольких передач, позволяющий платам сохранять за собой управление шиной и выполнять передачи без повторной процедуры арбитража. При передаче последовательных данных (т.е. данных, хранящихся в ячейках с последовательными адресами, например графических данных, дисковых файлов и т.п.) адрес следующего элемента при блочной передаче можно вычислить (а не передавать) одновременно с чтением или записью текущих данных. Если данные не находятся в последовательных ячейках, то требуется дополнительное время для арбитража и установки их адреса.

Во-вторых, можно передавать последовательные данные по адресным линиям, что позволяет удвоить пропускную способность. Наборы микросхем обладают буферами большей глубины, что позволяет объединять записи байтов, слов и двойных слов в единую операцию, т.е. выполнять меньшее число обращений в память.

В-третьих, реализуется пассивное освобождение шины, т.е. допускаются чередующиеся обращения к шине от процессора и других устройств даже в случае захвата шины транзакцией.

В-четвертых, можно буферизовать данные в контроллере шины PCI, что позволяет одновременно читать данные из памяти и осуществлять блочный обмен данными с ПУ.

В настоящее время шина PCI является стандартной, т.е. независимой от процессоров. Ее наличие стало стандартом де-факто для любых персональных компьютеров. Локальная шина PCI обеспечивает единообразное подключение всех ПУ и тем самым позволяет избежать «узких мест». Кроме того, эта шина до некоторой степени обеспечивает обратную совместимость с ПУ, предназначенными для подключения к шине ISA. Она создает некоторый промежуточный уровень между шиной процессора и периферийными устройствами. Стандарт PCI предусматривает обширный

перечень дополнительных функций. К ним относится автоматическая конфигурация ПУ, позволяющая пользователю устанавливать дополнительные платы контроллеров, не задумываясь о распределении прерываний, адресного пространства и каналов доступа в память.

Шина PCI служит для передачи 32- или 64-разрядных слов. Если по шине передаются только 32-разрядные слова данных, то она содержит 49 информационных линий; при расширении шины до 64 бит добавляется еще 48 линий. По этим линиям передаются 32- и 64-разрядные данные, а дополнительные линии служат для передачи контрольных разрядов, что позволяет исключить ошибки. К этой шине одновременно можно подключать до десятка разных ПУ, в том числе адаптеры локальных сетей и шин ISA и SCSI. Предусматривается групповой режим обмена при выполнении операций чтения и записи. В шине PCI применяется мультиплексирование, позволяющее передавать по одной электрической линии более одного сигнала. Благодаря этому сокращается число выводов адаптера ПУ и снижается его цена.

Основными преимуществами шины PCI являются:

поддержка синхронного обмена 32 или 64 бит данных с мультиплексированием передачи адресов и данных по одним линиям;

возможность установки компонентов с уровнями сигналов питания 3,3 или 5 В;

комбинирование частот шины 33 или 66 МГц с различной разрядностью данных, которое предоставляет широкий диапазон для выбора пропускной способности шины;

возможность передачи методом линейных пакетов (данные при записи-чтении передаются единым пакетом — нет необходимости передавать последовательные адреса);

использование различных способов кэширования.

Однако существенные ограничения по пиковой пропускной способности (30...40 Мбайт/с) при тактовой частоте шины 33 МГц стали замедлять рост производительности компьютерной системы. В частности, появление жестких дисков большого объема, сетевых карт, адаптеров SCSI потребовало увеличения пропускной способности шины PCI в несколько раз.

Для совершенствования шины была принята спецификация PCI-X, которая предполагает передачу 64-разрядных слов данных, тактовую частоту 133 МГц и передачу данных по протоколам DDR и QDR. В основном такая шина служит для высокопроизводительных серверов и рабочих станций, так как увеличение ширины шины (150 контактов в разъеме) и ее рабочих частот приводит к значительному удорожанию системной платы и компьютера в целом.

Основными претендентами на замену шины PCI являются интерфейс PCI Express (3GIO), разработанный фирмой Intel, и шина HyperTransport, предлагаемая фирмой AMD.

Шина AGP. Вскоре после внедрения шины PCI обнаружилось, что она не удовлетворяет возросшим требованиям при передаче графической информации. Поэтому для передачи видеоинформации было предложено использовать специальную шину, получившую название AGP (Accelerated Graphics Port). Ее главным преимуществом стала очень высокая пропускная способность — в режиме передачи 32-битных слов ее пиковая пропускная способность достигает 2132 Мбайт/с.

При разработке интерфейса AGP стремились разрешить две проблемы, связанные с особенностями обработки трехмерной графики на персональном компьютере. Во-первых, трехмерная графика требует выделения памяти для хранения текстур (стиля закрашивания) и Z-буфера (буфера глубины, предназначенного для удаления невидимых поверхностей). Использование для этих целей части ОП ограничивается пропускной способностью шины PCI. Во-вторых, шина AGP служит для прямого соединения графической подсистемы и контроллера ОП, тем самым разделяется доступ к ОП со стороны графической подсистемы и ЦП. Через эту шину AGP возможно подключение только графических плат.

В шине AGP использованы раздельные линии адреса и данных и, кроме того, введена конвейеризация операций чтения-записи, позволившая устранить влияние задержек в памяти на скорость их выполнения. Эта шина способна передавать два, четыре или восемь блоков данных за один цикл. Для контроллера AGP конкретный физический адрес, где хранятся данные в ОП, не имеет значения.

Шина AGP может работать в двух режимах: DIME и DMA. В режиме прямого доступа в память DMA текстуры могут храниться в ОП, но перед использованием они должны быть скопированы в локальную видеопамять. В режиме непосредственного исполнения DIME текстуры не копируются в видеопамять, а выбираются непосредственно из системной ОП. Системная память динамически осуществляет выделение блоков по 4 Кбайт, так как она может использоваться не только контроллером AGP, а для обеспечения приемлемого быстродействия предусмотрен специальный механизм, отображающий последовательные адреса на реальные адреса в ОП. Этот механизм основан на специальной таблице, хранящейся в ОП.

Операции шины AGP разделены, т.е. запрос на выполнение операции отделен от пересылки данных. Это значит, что можно генерировать запросы, не дожидаясь окончания текущей операции.

Интерфейс HyperTransport. Интерфейс HyperTransport (HT) предназначен для использования в высокопроизводительных системах (ВС) в качестве внутренней локальной шины. По сравнению с шиной PCI он позволяет уменьшить число проводников на

системной плате, устранить задержки, связанные с монополизацией шины устройствами с низкой производительностью, снизить энергопотребление и в целом многократно повысить пропускную способность системы.

Интерфейс HT организован на следующих уровнях:

физический — шина представлена 32 линиями, контроллерами и стандартными электрическими сигналами;

передача данных — определяет порядок инициализации и конфигурирования устройств, установления и прекращения сеанса связи, помехоустойчивого циклического кодирования и пакетирование данных;

протокол — определены команды выделения виртуальных каналов связи и правила управления потоком данных;

транзакции — команды протокола конкретизированы в управляющие сигналы, например чтения или записи;

сессии — определены команды управления энергопотреблением и другие команды общего характера.

Физические устройства в рамках интерфейса HT подразделяют на несколько типов: Cave («пещера») — оконечное устройство дуплексного канала связи; Tunnel («туннель») — промежуточное устройство дуплексного канала связи, отличное от моста; Bridge («мост») — устройство дуплексного канала связи.

Основными характеристиками интерфейса являются:

асинхронная последовательная пакетная передача по принципу «точка-точка»;

пиковая пропускная способность — 6,4 Гбайт/с при тактовой частоте шины 800 МГц (по обоим фронтам импульсов — 1600 МГц) и ширине шины 32 бита;

динамическое распределение линий.

С точки зрения схемотехнической организации шины HT принципиальным является ее масштабируемость. В минимальной конфигурации (ширина шины 2 бит, на каждый бит — две физические линии) требуется 24 контакта, а в максимальной (ширина канала 32 бит) — 197. В интерфейсе HT используются сигналы уровня 1,2 В (а не 2,5 В), «улучшенные» сигналы позволяют достичь длины шины около 60 см при скорости передачи данных до 800 Мбит/с.

Интерфейс PCI Express (3GIO). Этот интерфейс имеет топологию звезды и используется в качестве внутренней шины. Стандартизация процедур в этом интерфейсе осуществляется на нескольких уровнях взаимодействия.

Связь между устройствами производится по последовательным низковольтным дифференциальным линиям, по одной паре для передачи и приема данных. Масштабируемость шины достигается динамическим увеличением требуемого числа линий в 2, 4, 8, 16 или 32 раза. Между устройствами, участвующими в обмене дан-

ными по шине PCI Express, устанавливается выделенный канал связи, ширина которого и тактовая частота оговариваются устройствами в процессе инициализации канала. Тем самым реализуется процедура обмена типа «точка-точка», одновременно определяется формат пакетов (8 или 10 бит).

Аппаратно шина PCI Express управляется контроллером, получившим название Host Bridge. На уровне данных формируются подлежащие передаче пакеты и осуществляется помехоустойчивое кодирование. На уровне транзакций определяется готовность буфера приемного устройства, пересылается пакет, проверяется его получение, а при обнаружении ошибок осуществляется его повторная передача.

Логической частью интерфейса предусмотрена возможность передачи потока данных как по одному, так и по нескольким каналам. Такая организация обмена позволила обеспечить пропускную способность до 2,5 Гбит/с в каждом направлении, а с переходом на медные технологии увеличить ее до 10 Гбит/с. Предполагается, что интерфейс PCI Express заменит широко используемую в настоящее время шину AGP.

7.4. Интерфейсы внешней памяти

Развитие интерфейсов НЖМД шло двумя путями. Одно решение заключалось в создании в самом накопителе отдельного «интеллектуального» контроллера, который брал бы на себя значительную часть работы по взаимодействию с НЖМД. Результатом этого подхода явился интерфейс SCSI (Small Computer System Interface), быстро завоевавший популярность для серверов.

Более дешевое решение состояло в том, чтобы переложить значительную часть операций по управлению вводом-выводом на ЦП. Это решение имеет вполне очевидный недостаток: снижение мощности ВС, особенно заметное при многозадачной работе. Результатом такого подхода явился интерфейс IDE/ATA (Integrated Drive Electronics/AT Attachment for Disk Drives).

Интерфейс IDE (ATA). В выпущенном в 1980-х гг. компьютере PC AT накопитель на жестком диске был подсоединен к 16-битный шине ISA и управлялся отдельным собственным контроллером. Но затем было предложено встроить управляющую электронику в сам накопитель. Стандарт на такой интерфейс получил название ATA; он обеспечил возможность модернизации персональных компьютеров путем простой замены или добавления жестких дисков. Позднее появилось обозначение этого же интерфейса IDE.

Спецификация ATA определила, что к одному каналу можно подключать два устройства — ведущее и ведомое и установила

режимы программного обмена данными PIO и прямого доступа в память DMA.

Режим программного ввода-вывода PIO предусматривает участие ЦП в обмене данными между диском и ОП. В режиме прямого доступа в память DMA устройство напрямую общается с системной памятью, перехватывая управление шиной.

Интерфейс ATA разрабатывался исключительно для подключения жестких дисков. Между тем появились новые типы накопителей: дисководы, например CD-ROM, CD-RW, стримеры и т.д., кроме того, скорость считывания жестких дисков быстро увеличивалась и режимы, предусмотренные ATA, не удовлетворяли возрастающим требованиям. Все это вызвало появление нового интерфейса ATA-2.

Стандарт ATA-2 установил более быстрые протоколы PIO и DMA, определил новый режим обмена данными при передаче блоками, а также новую адресацию дискового пространства LBA (Logical Block Addressing). Затем в очередном стандарте ATA-3 была определена новая технология SMART (Self-monitoring Analysis and Reporting Technology). Существенным шагом вперед стало появление протокола ATAPI (ATA Packet Interface — пакетный интерфейс ATA). Он обеспечил возможность подключения к IDE компонентов, отличных от жестких дисков. Различные фирмы производители жестких магнитных дисков (ЖМД) стали использовать собственные названия этих интерфейсов.

Обобщенно возможности разновидностей интерфейса IDE/ATA представлены в табл. 7.1. Возможности дальнейшего совершенствования параллельного интерфейса IDE несмотря на появление жестких дисков UltraATA-133 практически исчерпаны.

Интерфейс SCSI. В отличие от IDE интерфейс SCSI (Small Computer System Interface — интерфейс малых вычислительных систем) разрабатывался для подключения не только дисков, а для восьми самых различных ПУ. Это универсальный интерфейс. Одно из этих устройств выполняет функции адаптера (host), т.е. связующего звена между шиной SCSI и системной шиной персонального компьютера. Шина SCSI взаимодействует не с самими устройствами, например дисковыми накопителями, а со встроенными в них контроллерами.

В интерфейсе SCSI предусматривается параллельная передача данных по 8, 16 и 32 линиям данных. Первый вариант SCSI имел 8-битную шину, данные по которой передавались со скоростью до 5 Мбайт/с. В настоящее время скорость обмена через интерфейс SCSI Ultra320 составляет до 320 Мбайт/с (а скорость передачи информации по шине IDE не превышает 133 Мбайт/с).

Стандартом SCSI определяются физические и электрические параметры параллельной шины ввода-вывода. Все устройства подключаются к интерфейсу SCSI в виде цепочки, а взаимодей-

Таблица 7.1

Основные возможности разновидностей интерфейса IDE/ATA

Параметры	ATA	ATA-2	ATA-3	ATA/ ATAPI-4	ATA/ ATAPI-5	ATA/ ATAPI-6
Режим работы	PIO 1	PIO 4 DMA 2	PIO 4 DMA 2	PIO 4 DMA 2 UDMA 2	PIO 4 DMA 2 UDMA 4	PIO 4 DMA 2 UDMA 5
Пропускная способность, Мбайт/с	4	16	16	33	66	100
Число подключаемых устройств	2	2	2	2 на канал	2 на канал	2 на канал
Число проводников шлейфа	40	40	40	40	80	80
Контроль четности	Нет	Нет	Нет	Есть	Есть	Есть

ствие между устройствами осуществляется по принципу «точка-многоточка». Каждое устройство в цепочке имеет уникальный идентификационный (ID) номер в диапазоне от 0 до 7 (в последних модификациях, допускающих подключение 31 устройства, от 0 до 31). Номер 7 по умолчанию присвоен SCSI хост-адаптеру. Обмен осуществляется по 50-проводному (или 68-проводному) кабелю длиной до 3 м. Может использоваться 16-разрядная (Wide) шина и быстрый обмен (Fast) данными. Проверка правильности передаваемой информации осуществляется контролем по четности. Чтобы упростить подключение и настройку НЖМД, накопителей на CD-ROM и ряда других устройств, для шины SCSI выпускаются контроллеры Plug-and-Play, позволяющие автоматически настраивать программы-драйверы и системные ресурсы.

Данные по шине SCSI передаются в синхронном или асинхронном режимах. Режим синхронизации устанавливается контроллером шины после обмена специальными сообщениями между двумя устройствами. Если оба устройства способны передавать (и принимать) данные в быстром синхронном режиме, то именно такой режим и будет установлен. При работе в асинхронном режиме получатель подтверждает получение каждого байта, а в синхронном — только целого пакета данных. Различные варианты шины SCSI различаются ее шириной, тактовой частотой, физическим типом интерфейса подключения. Все подключаемые к шине

SCSI устройства должны использовать дифференциальные сигналы низкого уровня.

Максимальная длина кабеля шины SCSI, составляющая 3 м, недостаточна для подключения НЖМД, расположенных в разных шкафах. Кроме того, за последние годы увеличились скорость передачи данных через интерфейс и число подключаемых устройств, а пропускная способность интерфейса оказалась недостаточной.

Интерфейс Fibre Chanel. Сравнительно недавно появился интерфейс, называемый Fibre Channel (волоконно-оптический канал), способный заменить SCSI. Первоначально предполагалось использовать его для организации высокоскоростной связи типа «система—система» или «система—подсистема» с использованием оптоволоконного канала. Долгое время применение Fibre Channel в качестве интерфейса внешних устройств считалось неоправданным из-за его высокой стоимости и относительно большого энергопотребления. Затем он был доработан для использования электронной (неоптической) транспортной среды и стал обеспечивать соединение различных ПУ (в том числе НЖМД) с компьютером. Спецификация FC-AL позволила использовать Fibre Channel в качестве стандартного интерфейса для НЖМД.

В отличие от SCSI, интерфейс Fibre Channel соединяет устройства не по единой шине, а по кольцевому каналу. При этом в кольцо могут быть объединены до 126 НЖМД и хост-устройств (в любой комбинации). Благодаря наличию специальных схем в интерфейсе Fibre Channel возможно «горячее» (т.е. без отключения питания) подключение НЖМД без нарушения нормальной работы канала.

Интерфейс FC-AL имеет следующие технические характеристики:

число подключаемых устройств — до 126;

максимальная скорость передачи — 100 Мбайт/с (при частоте 1,062 ГГц);

тип кабеля — двойной, коаксиальный, оптический;

особенности — возможность горячей замены, два порта подключения.

Однако несмотря на большие возможности интерфейса Fibre Channel FC-AL до сих пор НЖМД в персональных компьютерах и серверах чаще всего подключаются через интерфейсы SCSI и IDE.

7.5. Малые интерфейсы

К числу малых относят интерфейсы для подключения отдельных ПУ — накопителей на оптических дисках, принтеров, сканеров, модемов и т.д.

К параллельному интерфейсу долгое время было принято подключать принтеры, сканеры и другие типы устройств с достаточно высокой скоростью обмена. Но в последнее время все чаще стали использовать универсальный последовательный интерфейс USB, а также интерфейс IEEE-1394. К последовательным интерфейсам подключают все виды устройств, скорость работы которых по тем или иным соображениям принципиально ограничена. Большинство малых интерфейсов допускает подключение только одного устройства, поэтому в них не предусмотрена фаза адресации. Последовательные интерфейсы могут строиться по асинхронному и синхронному принципам.

Термин «последовательный» означает, что непосредственная передача символов осуществляется по одному проводу, при этом если в таком интерфейсе имеется несколько проводов, то они используются для функций управления, например запроса передачи, сигнализации готовности к приему и т.д.

Асинхронный интерфейс служит для последовательной передачи символов; примерно 20 % информации используется только для идентификации символа и осуществления синхронизации.

При синхронной передаче сообщения (синхронный интерфейс) передатчик поддерживает постоянные интервалы времени между выдачей очередных квантов информации. Принимающее устройство с помощью дополнительных сигналов, обеспечивающих синхронизацию, осуществляет прием передаваемых квантов в темпе их выдачи. В последовательность передаваемых байт передатчик вставляет коды символов синхронизации SYN; переход сигнала из состояния «0» в состояние «1» используется приемником для запуска своего генератора, работающего на той же частоте (таким образом осуществляется синхронизация). Приемник распознает передаваемый символ SYN и после этого принимает очередной байт сообщения. При невозможности распознать символ синхронизации, т.е. когда потеряна синхронизация, передатчик вставляет дополнительные символы синхронизации.

Интерфейс USB. Этот интерфейс сегодня является промышленным стандартом и рассчитан на подключение к компьютерам различных устройств бытовой электроники. Он предусматривал максимальную скорость передачи 12 Мбайт/с, но уже в момент его появления внутренние скорости чтения превышали: с флэш-памяти — 4 Мбайт/с, а с дисков Iomega Zip — 750 Мбайт/с. В связи с этим в 2000 г. вышла новая версия USB 2.0 с тактовыми частотами 1,5; 12 и 480 МГц. Теоретически максимальная скорость шины USB 2.0 составляет 60 Мбайт/с.

С точки зрения пользователя, для интерфейса USB характерны:

простота кабельной системы и возможность динамического «горячего» подключения и конфигурирования ПУ;

самоидентифицируемость ПУ, автоматический вызов драйверов и конфигурирование.

Шина USB позволяет адресовать до 127 устройств, подключаемых к компьютеру в виде «дерева». В его «корне» всегда находится активный контроллер компьютера, как правило, размещенный в концентраторе ввода-вывода. Все ресурсы шины распределяет центральный компьютер с помощью циклически повторяющихся кадров длительностью 1 мс.

Информационные сигналы и напряжение питания устройств 5 В передаются по четырехпроводному кабелю. Используется дифференциальный способ передачи сигналов по двум проводам. Интерфейс USB может работать в двух режимах передачи сигналов: с полной скоростью (12 Мбит/с для USB 1.1, 480 Мбит/с для USB 2.0) и низкой скоростью (1,5 Мбит/с). Для передачи с полной скоростью используется экранированная витая пара с длиной сегмента до 5 м, а для низкой — неэкранированный кабель длиной до 3 м. Низкая скорость предназначена для работы с небольшим числом сравнительно медленно действующих ПУ.

В процессе обмена все устройства «слушают», а центральный компьютер инициирует любую передачу в заданном направлении, монопольно посылая запросы в требуемое устройство. За запросами могут последовать данные.

В стандарте предусмотрено четыре типа обмена данными:

групповой обмен (Bulk), обеспечивающий надежную и скоростную передачу;

транзакция прерывания (Interrupt), позволяющая передавать данные по мере их готовности;

командный обмен (Control), используемый для управления устройствами;

обмен в изохронных потоках (Isochronous) для передачи синхронных по времени данных без гарантии достоверности их доставки.

Интерфейс USB обеспечивает одновременный обмен данными между центральным компьютером и множеством ПУ. Распределение пропускной способности шины между ПУ планируется компьютером и реализуется им с помощью посылки маркеров. Шина позволяет подключать, конфигурировать, использовать и отключать устройства во время работы компьютера и самих устройств.

В качестве устройств USB могут использоваться хабы (концентраторы), функции или их комбинации. Типичные функции — это отдельные ПУ с кабелем, подключаемым к порту хаба. В одном корпусе может находиться несколько функций, подключаемых с помощью встроенного хаба к одному порту, например различные указатели (мышь, планшет, световое перо), устройства ввода (клавиатура, сканер) и вывода (принтер, цифровые звуковые колонки), а также телефонный адаптер ISDN и т.д.

Точки подключения хаба называются *портами*. Каждый хаб преобразует одну точку подключения в множество точек. Архитектура допускает соединение нескольких хабов. У каждого хаба имеется один восходящий порт, предназначенный для подключения к компьютеру или хабу верхнего уровня. Остальные порты являются нисходящими, предназначенными для подключения функций или хабов нижнего уровня.

Хост делится на три части: интерфейсную, системную и программную. Каждая часть отвечает только за определенный круг задач.

В структуру USB входят следующие элементы:

физическое устройство USB — устройство на шине, выполняющее функции, интересующие конечного пользователя;

Client SW — ПО компьютера, соответствующее конкретному устройству;

USB System SW — системная поддержка USB, независимая от конкретных устройств;

USB Host Controller — аппаратные и программные средства для подключения устройств USB к компьютеру.

Каждое устройство USB представляет собой набор независимых конечных точек, с которыми контроллер обменивается информацией. Одна из этих точек имеет номер 0 и используется для инициализации, общего управления и опроса состояния контроллера.

Модель передачи данных между контроллером и конечной точкой (устройством) в интерфейсе USB называется *каналом*. Имеются два типа каналов: потоки и сообщения. Поток доставляет данные от одного конца канала к другому, он всегда однонаправленный и может реализовывать следующие типы обмена: сплошной, изохронный и прерывания. Выдача осуществляется в порядке «первым пришел — первым вышел» (FIFO). С точки зрения интерфейса USB, данные потока не структурированы.

Компьютер посылает запрос к конечной точке, после которого передается (принимается) пакет сообщений, за ним следует пакет с информацией о состоянии конечной точки. Последующее сообщение не может быть послано до обработки предыдущего, но при отработке ошибок возможен сброс необслуженных сообщений. Двухсторонний обмен сообщениями адресуется к одной и той же конечной точке. Для доставки сообщений используется только командный обмен.

Каналы организуются при конфигурировании устройств USB. Для каждого включенного устройства существует канал управления, по которому передается информация конфигурирования, управления и состояния. Интерфейс USB поддерживает как однонаправленные, так и двунаправленные режимы связи.

Архитектура USB допускает четыре базовых типа передачи данных:

1) управляющие посылки — пакетные, апериодические, используемые для конфигурирования и управления устройствами. Протокол обеспечивает гарантированную доставку данных. Длина поля данных управляющей посылки не превышает 64 байт при полной скорости и 8 байт при низкой;

2) большие массивы передачи данных — апериодические посылки больших пакетов без жестких требований ко времени доставки. Посылки занимают всю свободную полосу пропускания шины и допускаются только на полной скорости передачи. Пакеты имеют поле данных размером 8, 16, 32 или 64 байт. Приоритет этих пакетов самый низкий. Надежность обмена обеспечивается обнаружением ошибок и автоматической повторной передачей поврежденных данных;

3) прерывания — короткие (до 64 байт при полной скорости и до 8 — при низкой) пакеты. Прерывания должны обслуживаться со скоростью, определяемой устройством;

4) изохронные или потоковые передачи — непрерывные передачи в реальном времени, занимающие согласованную часть пропускной способности шины и имеющие заданную задержку доставки. При обнаружении ошибки поврежденные пакеты игнорируются.

Все транзакции по шине USB состоят из трех пакетов. Каждая транзакция начинается по инициативе контроллера, посылающего пакет-маркер, в котором указываются: тип и направление передачи, адрес устройства USB и номер конечной точки. Адресуемое маркером устройство распознает свой адрес и готовится к обмену.

Источник данных (определенный маркером) передает пакет данных (или уведомление об отсутствии данных, предназначенных для передачи). После успешного приема пакета приемник данных посылает пакет подтверждения.

Маркеры отвергнутых транзакций повторно передаются в свободное для шины время. Коды CRC позволяют обнаруживать все одиночные и двойные битовые ошибки, при этом контроллер автоматически производит трехкратную попытку передачи. Если повторы безуспешны, сообщение об ошибке передается клиентскому программному обеспечению.

Байты посылаются по шине последовательно, первым передается младший бит. Все посылки организованы в пакеты. Каждый пакет начинается с поля синхронизации. Последние два бита синхронизации служат маркером начала пакета. Идентификатором пакета является 4-битное поле, определяющее тип пакета, за которым в качестве контрольных следуют те же 4 бита, но инвертированные. Формат кадра USB 2.0 представлен на рис. 7.8.

Поле данных может иметь размер от 0 до 1023 байт. Размер поля зависит от типа передачи и согласуется при установлении канала.

Поле синхронизации канала	Поле идентификации пакета		Поле адресации	Поле данных
Sync	PID_{0-3}	$\overline{PID_{0-3}}$	A_{0-6}	D_{0-7}

Рис. 7.8. Формат кадра USB 2.0

Сигналы синхронизации и данных предаются по линиям связи, закодированными методом NRZI.

Интерфейс IEEE-1394 (FireWire). Последовательный интерфейс FireWire был впервые применен фирмой Apple. В дальнейшем он был переименован в интерфейс IEEE-1394 (рис. 7.9). Тактовая частота этого интерфейса составляет 98,304; 196,608 и 393,216 МГц, что соответствует скоростям передачи 100, 200 и 400 Мбит/с. Подключаемое к шине устройство может иметь любую максимальную скорость из этого набора, но обязано поддерживать и более низкую скорость.

Стандартный шестижильный экранированный кабель длиной до 4,5 м содержит две витые пары проводов, по которым осуществляется обмен данными, и два провода для питания устройств.

Каждое устройство является узлом сети (Node) и может содержать несколько равноправных разъемов (портов) для подсоединения кабеля. Между портами сигналы передаются через повторители. Такое соединение гарантирует устойчивость работы, исключает петли и упрощает процедуру арбитража. Одна сеть может содержать до 63 узлов, но сети могут объединяться.

Первоначально все узлы сети равноправны, а ее топология не определена. Любое изменение в сети переводит шину в состояние Reset. Затем следует идентификация, которая начинается с про-

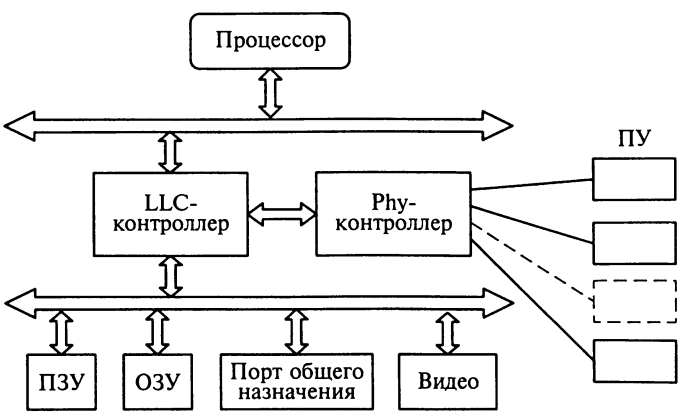


Рис. 7.9. Интерфейс IEEE-1394

цесса определения топологии и заканчивается самоидентификацией. В результате сеть приобретает топологию «дерева», а узлы получают уникальные адреса. Пока длится инициализация и идентификация, передача данных невозможна.

Согласование работы узлов обеспечивают специальные менеджеры. За администрирование шины IEEE-1394 отвечает менеджер шины (Bus Manager), менеджер питания (Power Manager) учитывает потребителей питания, а менеджер изохронных пакетов (Isochronous Manager) регулирует трафик.

В стандарте IEEE-1394 предусмотрена многоуровневая иерархия процедур. На физическом уровне выполняются действия, связанные с идентификацией, арбитражем и кодированием сигналов, на уровне соединения контролируется корректность пакетов и подтверждений, а на уровне пересылок обеспечивается правильный порядок пересылки пакетов. Средства уровня управления реализуют функциональность и обслуживают пакеты, предназначенные менеджерам.

Функционирование шины IEEE-1394 основывается на «честном» поведении узлов сети, т.е. все узлы должны быть исправны, не перехватывать и не отвечать на чужие пакеты и занимать шину только в установленное для этого время.

Изохронный (синхронный) цикл длится 125 мкс и начинается со специального 20-байтного пакета — *метки начала цикла*. Устройство, желающее передать данные в изохронном пакете, резервирует канал и полосу пропускания (не более 80 % времени цикла). Изохронные пакеты должны следовать один за другим в соответствии с номером канала сразу после метки начала цикла. От устройства требуется, чтобы размер его пакетов не превысил зарезервированную полосу. Данные, содержащиеся в изохронных пакетах, могут одновременно быть принятыми несколькими устройствами. Объем передаваемых данных в одном изохронном пакете достигает 4096 байт (при скорости 400 Мбит/с). Заголовок с уникальным номером канала и проверочные суммы занимают 12 байт, поэтому эффективность изохронных пакетов высока. Использование изохронных пакетов целесообразно, например, для передачи аудио- и видеоинформации, для которой незначительные потери несущественны.

Цикл при асинхронном обмене имеет произвольную длительность, зависящую от числа активных пользователей, желающих одновременно передать пакет. Устройства передают свои пакеты по очереди в соответствии с приоритетом, определяемым топологией сети. Узел, не сумевший передать пакет, может выполнить повторную передачу, если в течение некоторого интервала никто не «захватил» шину. Как правило, на любой пакет запроса должен поступить пакет ответа. Запросы бывают трех видов: чтения (Read), записи (Write) и комбинированной операции (Lock). В заголовке

асинхронного пакета передается адрес источника, адрес приемника, тип операции, размер пересылаемых данных и глобальный адрес в адресном пространстве устройства. В результате размер заголовка вместе с проверочной суммой может достигать 20... 24 байт.

Недостаток таких пакетов — большие заголовки. Если учесть, что весь цикл чтения состоит из запросов и ответов со своими заголовками, то на одну пересылку приходится 48 байт служебной информации. Даже при размере поля данных пакета в 32 байта эффективная скорость более чем в два раза меньше максимальной. Эффективность обмена возрастает при увеличении размера пакета. Для управления и контроля используются исключительно асинхронные пакеты.

Управление IEEE-1394 осуществляется двумя (реже одной) интегральными схемами — контроллером физического уровня Phy и контроллером уровня соединения LLC.

Контроллер физического уровня Phy реализует все процедуры физического уровня: прием-передачу аналогового сигнала, контроль состояния линии, формирование-контроль временны́х интервалов, кодирование-декодирование данных, арбитраж и т.д.

Контроллер уровня соединения LLC обеспечивает связь между контроллером физического уровня, шиной компьютера (например, PCI) и устройствами. Он контролирует прием-передачу пакетов между Phy-контроллером и внутренними регистрами. Контроллеры FireWire в настоящее время обычно устанавливаются на системную плату дополнительно.

При рассмотрении интерфейсов нужно обратить внимание на следующее:

за счет чего достигается высокое быстродействие, и чем оно принципиально ограничено для данного интерфейса;

сколько устройств можно подключать к интерфейсу, и чем лимитируется число подключаемых к нему устройств;

как распределяется время на обслуживание устройств, т.е. каким образом выполнена схема арбитража.

Многие широко распространенные в прошлом интерфейсы сегодня не используются, так как не обеспечивают нужной скорости передачи информации или не позволяют подключить нужное число устройств. Однако в персональных компьютерах прежних выпусков могут встречаться интерфейсы Centronics, RS-232C и ряд других.

Контрольные вопросы

1. Дайте определение интерфейса. Каково место интерфейса в системе передачи информации? Что означают понятия «порт», «шина», «магистраль», «линия и канал связи»?

2. Какие виды интерфейсов используются в различных вычислительных системах и персональных компьютерах?

3. Какие подшины образуют полную шину компьютера? Какие подшины могут быть совмещены? Что дает такое совмещение?

4. Чем характеризуется шина? Какой пропускной способностью обладают известные вам шины?

5. В чем заключается сущность симплексного, полудуплексного и дуплексного способов передачи сигналов?

6. Как производится арбитраж шины? Что такое арбитраж, какие функции на него возлагают?

7. Каким образом осуществляется передача в последовательном асинхронном и синхронном интерфейсах?

8. Как проводится передача в параллельном интерфейсе? Что помогает избежать «перекоса» информации?

9. Какие методы используют для реализации синхронной и асинхронной передачи данных? Поясните на примере сущность и назначение стробирования.

10. Чем определяется максимальное число подключаемых устройств к интерфейсу?

11. Изобразите представление одно- и двухполярных низкочастотных сигналов. Какими параметрами принято характеризовать одиночный импульс?

12. Какой пропускной способностью обладает шина PCI? Какое число бит одновременно можно передавать по ней?

13. Какие функции возлагают на набор микросхем (chipset) шины PCI? Какие наборы вам известны?

14. Почему в компьютерах используют несколько интерфейсов для подключения ПУ?

15. Сколько ПУ и с какой скоростью могут работать в интерфейсе ISA?

16. Какими характеристиками обладает интерфейс USB? Какие устройства к нему подключают?

17. Какие особенности имеет интерфейс SCSI?

18. Какой пропускной способностью обладает интерфейс SCSI различных модификаций?

19. Что такое технология Plug-and-Play? Любое ли устройство можно подключить к интерфейсу, чтобы оно отвечало требованиям этой технологии?

20. Существуют ли ограничения на число устройств, которые можно подключать к интерфейсу SCSI?

21. Проведите сравнительный анализ интерфейсов IDE (ATA) и SCSI.

22. Проведите сравнительный анализ интерфейсов HyperTransport и PCI Express.

23. Проведите сравнительный анализ интерфейсов USB и IEEE-1394 (FireWire).

24. В чем состоят особенности, достоинства и недостатки интерфейсов Serial ATA и Fiber Channel?

ГЛАВА 8

ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА КОМПЬЮТЕРОВ

8.1. Организация систем ввода-вывода. Каналы, контроллеры

Устройства ввода-вывода информации служат для взаимодействия компьютера с различными источниками и потребителями информации. Внешний мир характеризуется разнообразием объектов и различными формами представления информации: текстовой, графической, речевой, в виде аналоговых или дискретных сигналов. Обмен сообщениями между центральной частью машины и объектами внешнего мира осуществляется системой ввода-вывода. Такая система включает в себя ПУ, служащие для преобразования и кодирования информации, а также разнообразные средства управления обменом и передачи сообщений в компьютер и из него.

Современные компьютеры позволяют подключать к ним самые разнообразные ПУ, т.е. представляют собой системы с переменным составом оборудования. Для их реализации необходима стандартизация форматов передаваемых сообщений и алгоритмов управления аппаратурой, а также наличие универсальных программных модулей для управления ПУ. В зависимости от назначения и мощности компьютера системы ввода-вывода (СВВ) имеют различные структуры, а для их построения используются разные сочетания аппаратных и программных средств.

Основные функции СВВ. Задача СВВ состоит в организации и управлении процессом передачи информации между ПУ и ОП компьютера. Для СВВ любое ПУ представляет собой своеобразный генератор (устройства ввода) или потребитель (устройства вывода) данных D_i, работающий под управлением сигналов C_i и сообщающий о своем состоянии СВВ сигналами S_i, (рис. 8.1).

На СВВ возлагаются следующие функции:

формирование текущего адреса A_i ОП и передачи данных по этому адресу;

выработка управляющих сигналов C_i;

получение и обработка сигналов состояния S_i и формирование сообщений о состоянии СВВ;

получение приказов от ЦП на выполнение операций ввода-вывода.

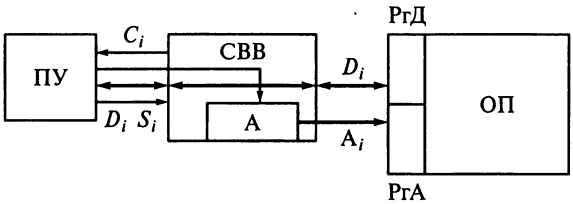


Рис. 8.1. Основные функции СВВ

Кроме того, СВВ обеспечивает синхронизацию процессов в ПУ и центральной части компьютера, согласование скоростей их работы, буферизацию данных и преобразование их форматов.

Такой большой набор функций может быть реализован самыми различными способами. В самых первых компьютерах эти функции выполнялись при помощи арифметического устройства и центрального устройства управления, поэтому совмещать операции ввода-вывода и обработки было невозможно. Но это приводило к существенному замедлению работы компьютера, так как операции ввода-вывода выполнялись с участием медленных ПУ. Затем были реализованы системы прерываний и приостановок, позволившие совместить во времени обработку и ввод-вывод. Система ввода-вывода образует маршрут передачи данных между ПУ и ОП и осуществляет управление обменом. Поскольку передача данных происходит по специальным каналам и в соответствии с определенными правилами, то в СВВ обязательно входят аппаратные средства.

Все функции СВВ можно разбить на три группы:

1) установление связи между ПУ и ОП;

2) определение текущего адреса ячейки памяти для записи или чтения очередного слова данных;

3) разрушение установленной связи и возвращение всех компонентов системы в исходное состояние.

Во время установления связи образуется канал передачи данных между одним из ПУ и ОП, по которому и будут передаваться данные. Для этого выявляют наиболее приоритетный запрос и исключают возможность передачи от любого другого ПУ, проверяют работоспособность и готовность к работе всех соответствующих компонентов СВВ и передают им нужную управляющую информацию.

В выполнении всех перечисленных действий обязательно участвует процессор.

Определение текущего адреса, преобразование форматов передаваемых данных, контроль передачи и выявление особых условий может выполняться как средствами процессора (программный ввод-вывод), так и автономными аппаратными средствами

(прямой доступ в память). Это наиболее длительная часть операции ввода-вывода.

Третья группа функций связана с завершением текущей операции ввода-вывода и «разрушением» канала. Здесь определяются момент и причины (нормальное завершение или завершение вследствие выявления ошибки) завершения обмена, а также выполняется передача управляющей информации компонентам СВВ для перевода их в исходное состояние.

Программный ввод-вывод. Определение текущего адреса данных, производится процессором программным путем. Если ПУ подготовило данные для передачи в память компьютера, то оно формирует сигнал прерывания и передает его в процессор. Этот сигнал переводит один разряд специального регистра в состояние «1». Выполнение каждой команды в процессоре завершается опросом этого регистра. При обнаружении сигнала прерывания происходит «переключение» процессора с текущей программы на программу обслуживания устройства, приславшего сигнал прерывания.

Программа обслуживания ПУ получает слово данных (обычно, байт), передаваемое устройством, и определяет адрес ячейки памяти, куда (или откуда) оно должно быть направлено. Эта программа, помимо определения адреса, осуществляет буферизацию, определяет число подлежащих передаче слов, выполняет контроль передачи и преобразование форматов. После завершения передачи одного или нескольких слов из ПУ в память компьютера происходит восстановление текущей программы процессора. Это и есть программный ввод-вывод.

При программном вводе-выводе не может быть обеспечена работа быстрых ПУ, например дисковых накопителей. Затраты времени на передачу одного байта из ПУ в память машины, т.е. на выполнение процессором функций «канала», даже при высоком быстродействии современных процессоров сравнительно велики, поэтому для обслуживания быстрых ПУ используют прямой доступ к памяти. Программный ввод-вывод характерен для персональных компьютеров и микроЭВМ при подключении к ним сравнительно медленных ПУ.

Прямой доступ в память. Для его реализации предусматривают специальные аппаратные средства, которые осуществляют наиболее затратные по времени функции, а именно буферизацию данных и преобразование их форматов, определение текущего адреса ОП, куда пересылаются байты данных, и числа еще не переданных байт. Остальные функции по установлению связи и образованию «канала» между ПУ и ОП и завершению операции выполняются как ЦП, так и специальными процессорами ввода-вывода.

Настройку канала прямого доступа в персональных компьютерах ЦП выполняет программно. Получив сигнал прерывания, он

переключается на выполнение программы-драйвера и передает в контроллер ПУ команду, начальный адрес, число подлежащих передаче байт, а также информацию о размещении этих байт на носителе (например, номер сектора на диске). После завершения передачи управляющей информации ЦП возвращается к первоначальной программе, а контроллер ПУ начинает процедуру прямого доступа в память, т.е. принимает передаваемый байт из ПУ (при вводе), передает этот байт в ячейку ОП по хранящемуся в регистре контроллера адресу, увеличивает этот адрес и уменьшает число байт, подлежащих передаче. Управление передачей каждого байта не требует вмешательства со стороны ЦП, а возможные конфликты при обращении к памяти разрешаются посредством приостановок. Процедура передачи байт в ОП завершится, как только счетчик байт обнулится. При прямом доступе в память возможно совмещение операций ввода-вывода и обработки, но только если в ЦП предусмотрена большая буферная память, так как в современных компьютерах устройства прямого доступа (например, дисковая память) обладают высоким быстродействием и при обращении к ним полностью захватывают магистраль для передачи данных.

В мейнфреймах и суперкомпьютерах все функции ввода-вывода выполняются специально выделенными средствами, работающими под управлением собственной программы. Эти средства образуют процессор ввода-вывода (ПВВ). Однако работа такого ПВВ подчинена программам ЦП: она инициируется по команде ЦП, после этого выполняется автономно до своего завершения, а после окончания работы ПВВ сообщает ЦП о том, как она завершилась. Наличие ПВВ позволяет существенно разгрузить ЦП от работ, связанных с вводом-выводом, но не от инициирования их.

8.2. Клавиатура и мышь

Клавиатуры предназначены для ввода буквенно-цифровой или текстовой информации в память компьютера, а мыши — для указания какой-либо определенной точки на экране, например некоторого окна. В переносных компьютерах вместо мыши используют различные трекболы, шаровые манипуляторы, джойстики и другие подобные устройства, служащие для управления специальной меткой — маркером, перемещаемым по экрану. Для ввода текста помимо клавиатуры могут использоваться также устройства автоматического распознавания печатных и рукописных символов, устройства ввода с промежуточного носителя (например, с магнитной ленты или дискеты) и т.п.

Клавиатура. Она состоит из совокупности ключей, замыкаемых при нажатии клавиш, а также схем управления для формирова-

ния кода символа, исключения неоднозначности кодирования из-за «дребезга» контактов и выполнения других управляющих функций. Традиционная клавиатура содержит более 100 клавиш. Число клавиш всегда меньше числа символов в алфавите, поэтому на клавиатуре располагают специальные управляющие клавиши, изменяющие коды остальных.

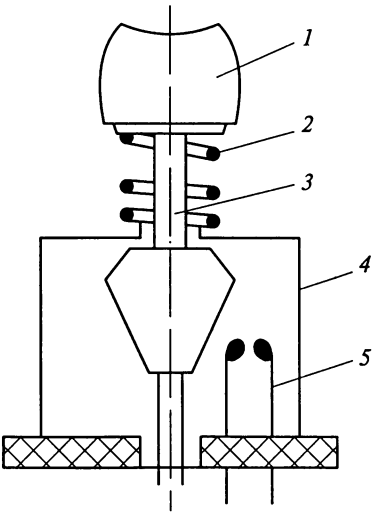


Рис. 8.2. Устройство ключа клавиатуры:

1 — клавиша; *2* — возвратная пружина; *3* — плунжер; *4* — корпус; *5* — контакты

В настоящее время существует несколько разновидностей ключей, но наиболее распространен ключ (рис. 8.2), состоящий из клавиши *1*, возвратной пружины *2*, плунжера *3*, корпуса *4* и собственно контактов *5*. При нажатии на клавишу контакты замыкаются, что приводит к изменению электрического сопротивления между ними. В некоторых клавиатурах вместо механических контактов используют подвижную и неподвижную пластины, при нажатии на клавишу между которыми происходит изменение емкости, или полупроводник, в котором возникает разность потенциалов под действием магнитного поля, создаваемого закрепленным на плунжере подвижным магнитом.

Клавиатура представляет собой прямоугольную матрицу, состоящую из ключей, расположенных на пересечении столбцов и строк (рис. 8.3). Генератор тактовых импульсов, счетчик и дешифратор служат для последовательного опроса состояния ключей в столбцах клавиатуры. Сигнал с дешифратора поступает на очеред-

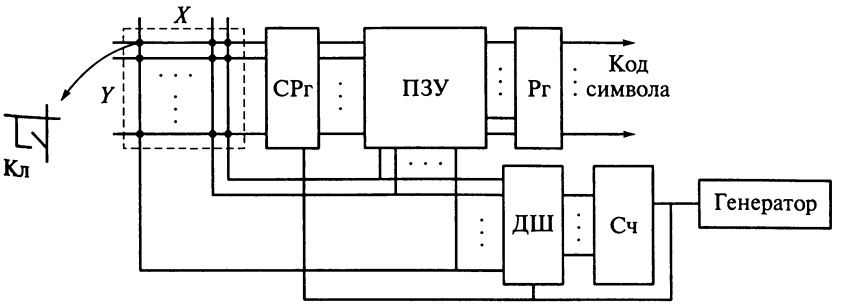


Рис. 8.3. Схема управления клавиатурой

ной столбец клавиатуры и соответствующий адресный вход ПЗУ (вход *X*). Если в данном столбце находится нажатая клавиша, то этот сигнал проходит и на второй адресный вход *Y*. В ячейках ПЗУ записаны коды символов, таким образом, содержимое ячейки с адресом *XY*, т.е. код нужного символа, выдается на выходной регистр. Поскольку число клавиш на клавиатуре меньше числа символов, то в ПЗУ записаны не полные коды символов, а лишь младшие разряды. Старшие разряды определяются содержимым специального регистра (СРг), состояние которого фиксируется при нажатии клавиш переключения регистров, например клавиши Shift. Для исключения влияния «дребезга» контактов выдача символа из регистра осуществляется с задержкой на время завершения переходного процесса.

Для управления клавиатурой можно использовать и микропроцессоры. В этом случае ее шины *XY* подключаются к портам ввода и вывода, а для передачи в компьютер сформированного программным путем кода символа используется второй порт вывода. При микропроцессорном управлении достаточно просто избежать влияния «дребезга» контактов и фиксировать ошибочные состояния, когда одновременно оказываются нажатыми несколько клавиш. Для этих целей программа определения кода символа выполняется несколько раз и производится сравнение полученных кодов. Передача кода символа в выходной порт осуществляется только в случае многократного совпадения полученных кодов.

Мышь. Снизу механической мыши (рис. 8.4) находится шар, который поворачивается при перемещении мыши по плоскости стола (или специального «коврика»). Шар касается двух расположенных под прямым углом друг от друга валиков *2*, *3*, которые при движении мыши также поворачиваются. Угол поворота валика пропорционален углу поворота шара и синусу угла между направлением перемещения мыши и осью вращения этого валика. Валики связаны с элементарными преобразователями угла поворота в цифровой код. Чаще всего такие преобразователи устроены в виде диска с прорезями *4*, полупроводникового излучателя и фотоприемников *5*.

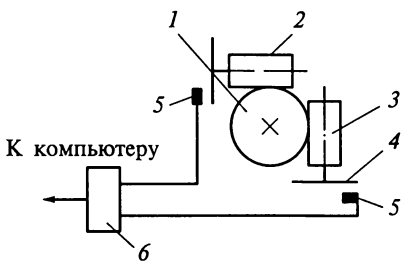


Рис. 8.4. Устройство мыши:

1 — шар; *2*, *3* — валики; *4* — диск с прорезями; *5* — фотоприемники; *6* — преобразователь

Каждый преобразователь имеет по два фотоприемника для определения направления перемещения шара. На выходах фотоприемников формируются импульсы, число которых пропорционально перемещению мыши. Полученные на выходе преобразователей *6* коды передаются в

компьютер и служат для пропорционального перемещения маркера по экрану.

Однако такая мышь часто начинает плохо работать из-за загрязнения валиков и возникающего проскальзывания, поэтому современные конструкции мыши отличаются от описанной выше. В последнее время получила распространение оптическая мышь, в которой вместо шара и преобразователей углов поворота в цифровые коды использован полупроводниковый красный лазерный излучатель. Луч света от этого излучателя направляется на поверхность, по которой осуществляется перемещение мыши, и, отражаясь от нее, попадает в объектив монохромной КМОП-камеры, которая фотографирует небольшой (около 1 мм^2) участок поверхности. *Фотографирование* — это процесс разбиения участка поверхности на небольшие квадратики, число которых может быть 16×16, и вычисления усредненного значения яркости для каждого из них. Фотографирование производится сенсором, расположенным за объективом камеры. Конструктивно оптический сенсор и объектив выполнены в виде одной интегральной схемы, на нижней стороне которой располагается объектив. Каждому квадратному участку присваивается значение яркости от 0 до 63, где 0 соответствует белому, а 63 — черному участкам. Таким образом, получается мозаичное цифровое описание участка поверхности.

Это цифровое описание участка изображения передается от сенсорного датчика в сигнальный процессор, где запоминается и сравнивается с предыдущим описанием. Если описания полностью совпадают, то мышь не перемещалась; если совпадения не выявлено вовсе, то мышь перемещалась слишком быстро. Если полного совпадения нет, но выявлена некоторая общая часть изображения, то сигнальный процессор определяет величину и направление перемещения мыши (это сделать несложно) и передает новые координаты мыши в компьютер. Поверхность фотографируется с очень большой скоростью — около 1500 снимков в секунду, что позволяет перемещать мышь довольно быстро со скоростью до 25 см/с. Оптическая мышь практически не загрязняется и позволяет работать достаточно быстро, но не на любой поверхности. Существуют оптические мыши, связанные с компьютером не проводами, а через инфракрасный порт. Однако такие мыши не получили особо широкого распространения из-за сравнительно высокой стоимости.

8.3. Дисплеи

Устройства оперативного отображения информации, или дисплеи, долгое время строились исключительно на основе электронно-лучевых трубок (ЭЛТ). Экран такого дисплея представляет

собой прямоугольную сетку, в узлах которой расположены отдельные элементарные индикаторы (ЭИ), изменяющие свои оптические свойства под воздействием управляющих сигналов. На экране ЭЛТ такая сетка может создаваться в процессе формирования изображения. Включение и выключение ЭИ для формирования видимого изображения производится в последовательности, называемой *опросом*.

В настоящее время используют *растровый метод*, при котором последовательность опроса всегда постоянна и не зависит от характера изображения. Луч (а точнее, три луча — красный, зеленый и синий, или RGB — для создания цветного изображения) равномерно перемещается по экрану слева направо, прочерчивая строку растра. Это прямой ход луча. Затем выполняется обратный ход луча, по окончании которого он занимает положение в начале следующей строки, т.е. происходит перемещение луча сверху вниз. После выполнения последнего прямого хода, когда луч достиг крайнего нижнего положения, осуществляется его возврат в начало. Этот полный цикл называется *кадром*.

Во время прямого хода интенсивность луча можно менять, что, в свою очередь, ведет к изменению интенсивности излучения строго определенной точки экрана. Но точка на экране светится недолго, и нужно поддерживать ее состояние, т.е. многократно повторять весь процесс, чтобы изображение стало видимым. *Частота повторения процесса*, или *частота кадров*, должна быть не меньше критической частоты мерцаний, в современных дисплеях она составляет 80...160 Гц.

Многократное повторение процесса отображения информации на экране ЭЛТ требует запоминания этой информации в буферной памяти. Если используется растровый метод формирования изображений, то объем такого буфера нетрудно подсчитать: он определяется произведением числа адресуемых точек на экране и числа бит, необходимых для записи параметров (например, цвета, яркости) каждой точки. Число адресуемых точек, а следовательно, и разрешающая способность экрана современного дисплея, составляет от 768×1024 до 1200×1600, а число бит для записи параметров — до 64. Таким образом, необходима буферная память объемом 64...128 Мбайт. Именно необходимость памяти большого объема не позволила создавать дисплеи до появления БИС. Первые дисплеи были алфавитно-цифровыми, позволяли формировать изображения ограниченного числа символов (описания всех отображаемых символов хранились в ПЗУ) и требовали буферной памяти сравнительно небольшого объема.

Растровые графические дисплеи, по сравнению с алфавитно-цифровыми, требуют значительно большей буферной памяти, обладающей высоким быстродействием. Эта буферная видеопамять должна хранить описания каждого графического примитива (эле-

ментарного графического изображения) в растровой форме, т.е. в виде отдельных точек, а не векторной, когда этот примитив обрабатывается в машине. Кроме того, графический дисплей позволяет производить процедуры над графическим примитивом: перемещать, поворачивать, масштабировать, стирать и т.д. Эти процедуры реализуются над примитивом, представленным в векторной форме. По этой причине в структуре дисплея должен быть предусмотрен блок (рис. 8.5), выполняющий функции векторного графического процессора (ВГП) и ОЗУ для хранения описания дисплейного файла (ОЗУ ДФ). Преобразованный в растровую форму посредством растрового графического процессора (РГП) файл сохраняется в памяти видеоконтроллера (ВК). Оператор взаимодействует с таким дисплеем посредством специального блока интерактивного взаимодействия (БИВ) с помощью клавиатуры, мыши и других средств.

Помимо формирования описания РГП совместно с ВК выполняет функцию кадрирования, т.е. формирует окно изображения. Процедура кадрирования может осуществляется программными и аппаратными средствами. Для аппаратного выполнения этой процедуры в видеоконтроллере предусматривают специальные счетчики размера окна.

Графические дисплеи имеют «жесткую» временну́ю диаграмму. Это значит, что время экспозиции точки растра всегда постоянно и определяется частотой регенерации кадра, числом строк в кадре и растровых элементов в строке. Эти параметры оказывают существенное влияние на преобразования, выполняемые растровым графическим процессором.

Недостатки дисплеев на ЭЛТ — большие габаритные размеры и масса, высоковольтное питание, сравнительно большое энергопотребление — давно уже определили их участь: они постепенно вытесняются плоскими мониторами на жидких кристаллах (ЖК). Однако и в ЖК-мониторах сохраняется необходимость векторно-растрового преобразования.

Принцип действия любого ЖК-монитора основан на способности некоторых типов жидкокристаллических веществ повора-

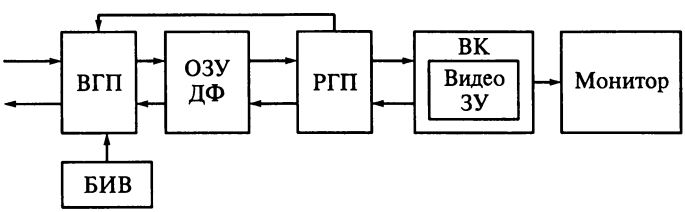


Рис. 8.5. Структура графического дисплея

чивать плоскость поляризации проходящего через них света, при этом положение кристалла можно менять электрическим полем.

Такая панель представляет собой две стеклянные пластины с нанесенными на них прозрачными электродами. Между стеклянными пластинами находятся ЖК, а сверху и снизу они покрыты поляризующими пленками, или поляроидами (рис. 8.6). Снизу эта конструкция подсвечивается ртутной флюоресцентной лампой с холодным катодом. Пусть обе пленки имеют горизонтальную поляризацию, а ЖК расположены так, что не меняют направление поляризации света, когда электрическое поле отсутствует. Тогда свет от лампы, пройдя сквозь первую поляризующую пластину, получит горизонтальную поляризацию, без изменений пройдет сквозь слой ЖК, а затем и через второй поляризатор. Такая панель будет казаться прозрачной.

Если же изменить положение ЖК (а для этого нужно создать электрическое поле) так, чтобы они поворачивали плоскость поляризации проходящего через них света на 90°, то свет не пройдет сквозь вторую поляризующую пластину и панель станет непрозрачной. (Свет будет полностью поглощен вторым поляризатором, так как направление его поляризации будет перпендикулярно направлению поляризации второй пластины.) Поворачивая жидкие кристаллы на промежуточные углы, можно плавно регулировать прозрачность панели. Для получения цветного изображения каждый пиксел панели разбивается на три субпиксела и на него накладывается цветоделительная маска, окрашивающая проходящий через каждый субпиксел свет в один из основных цветов: красный, синий и зеленый.

Однако ЖК-мониторы, в основу которых положена модуляция внешнего света, обладают низкой контрастностью. Это вызвано тем, что поляризаторы всегда пропускают пусть даже небольшую часть света, т.е. черный цвет не может иметь нулевую интенсивность свечения. Кроме того, из-за большой толщины

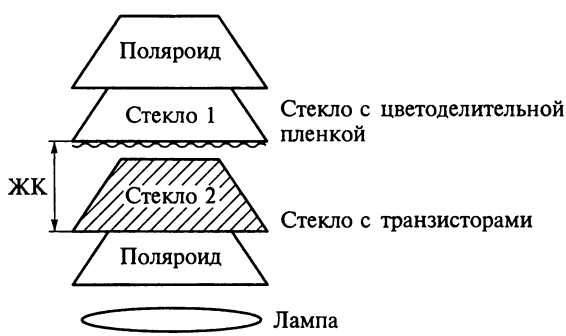


Рис. 8.6. Структура жидкокристаллической панели

панели угол обзора оставался крайне небольшим, а в силу большой длины электродов, подходящих к каждой точке, такие панели обладали весьма большим временем реакции. По этим причинам такие панели в настоящее время уже не используются, а вместо них выпускаются активно-матричные, в которых каждый пиксел имеет собственный управляющий транзистор и снабжен поддерживающим напряжение конденсатором. Такие панели называют активными матрицами, или *TFT-панелями*, так как в них используют тонкопленочные транзисторы (толщиной менее 0,1 мкм) для обеспечения прозрачности матрицы.

В ЖК-мониторе используется постоянная подсветка элемента экрана. Она позволяет избежать мерцаний. Управляющее напряжение приложено к каждой ячейке в течение кадра (в отличие от дисплеев на ЭЛТ, где применяется импульсный способ засветки элементов экрана), поэтому не требуется высокая частота регенерации. Однако длительное свечение точек изображения ухудшает воспроизведение быстродвижущихся объектов. По этой причине далеко не все ЖК-мониторы пригодны для динамичных игр и просмотра видеофильмов.

Размеры современных ЖК-мониторов составляют 15, 17, 19 и 26 дюймов, а стандартное разрешение 1280 × 1024 при размере пиксела в 0,264 мм. Они обладают достаточно высокой яркостью, а углы обзора по горизонтали и вертикали составляют порядка 150... 170°. Время переключения пиксела лежит в пределах от 16 до 25 мс. Благодаря компактности конструкции в ЖК-мониторах можно предусмотреть возможность поворота экрана на 90°, что удобно при работе с текстовой информацией. Кроме того, для ЖК-мониторов не требуется высокого напряжения, а потребление электроэнергии у них сравнительно невелико.

8.4. Принтеры

Обработанная на компьютере информация часто должны быть представлена в документированном виде на бумаге или ином подобном носителе. Для этого служат принтеры, или печатающие устройства. Основными требованиями, предъявляемыми к современным принтерам, являются: высокое качество печати, использование различных шрифтов, возможность печати графики, получения многоцветных изображений и нескольких копий, высокое быстродействие и низкая стоимость. Указанные требования могут противоречить друг другу, что приводит к большому разнообразию конструкций.

Все принтеры принято классифицировать по следующим признакам: способу регистрации изображения (лазерные, струйные, ударные и т.д.), способу формирования изображения (знакопеча-

тающие и матричные), числу одновременно формируемых символов (последовательные, построчные и страничные), в соответствии с размерами бумаги, на которой формируется изображение.

В настоящее время наибольшее распространение получили лазерные и струйные принтеры. Из-за шумности, недостаточно высокого быстродействия и невысокой надежности пользовавшиеся широким спросом матричные и знакопечатающие принтеры ударного типа уходят в прошлое. Принтеры, основанные на термографическом принципе формирования изображения, применяются при печати чеков, билетов и т. п. В них используется специальная термобумага, меняющая цвет при тепловом воздействии на нее.

Рассмотрим принципы лазерной и струйной печати. *Лазерные* печатающие устройства обладают высокой скоростью печати, достигающей 20...25 страниц в минуту, и обеспечивают высокое разрешение в 1200 точка/дюйм и выше. Обычно лазерные принтеры предназначаются для рабочих групп, но существуют модели и для индивидуального пользования. Если первые лазерные устройства обеспечивали исключительно монохромную печать, то в настоящее время существуют устройства, позволяющие получать цветные отпечатки. Лазерный способ регистрации изображения основан на явлении местного разрушения электростатического заряда, созданного в слое полупроводникового материала под действием света.

Структура одного из первых лазерных принтеров показана на рис. 8.7, *а*. Вращающийся барабан *4*, покрытый слоем фотопроводника, заряжается вследствие образования на его поверхности

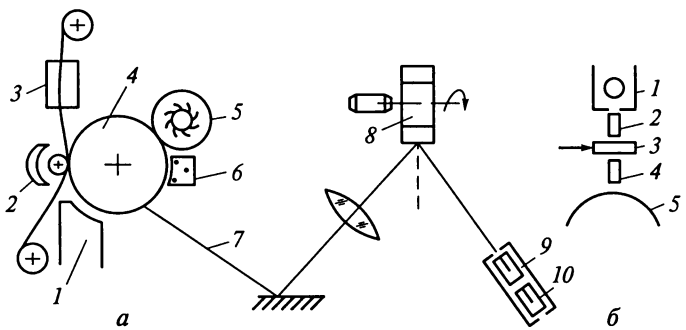


Рис. 8.7. Лазерный принтер:

а — формирование видимого изображения с помощью лазерного луча: *1* — краситель; *2* — бумага; *3* — устройство термосилового закрепления; *4* — барабан; *5* — устройство очистки барабана; *6* — ионизатор; *7* — луч; *8* — зеркало; *9* — модулятор; *10* — лазер; *б* — формирование видимого изображения газоразрядной лампой: *1* — газоразрядная лампа; *2* — система световодов; *3* — панель ЖК-затворов; *4* — фокусирующая матрица; *5* — барабан

слоя положительных ионов под действием ионизатора *6*. Ионизатор заряжает всю поверхность барабана. При попадании на эту поверхность луча света *7* происходит точечное разрушение заряда. Таким образом, если избирательно осветить поверхность барабана, то на ней можно создать скрытое электростатическое изображение, формируемое лазерным лучом лазера *10* и модулируемое модулятором *9*. Перемещение луча по поверхности барабана осуществляется многогранным зеркалом *8*. Скрытое электростатическое изображение проявляется путем осаждения на положительно заряженные участки поверхности отрицательно заряженных частиц красителя *1*. Затем проявленное изображение переносится на бумагу *2*, где краситель фиксируется путем термосилового закрепления *3*. Поверхность барабана *4* очищается для печати следующей страницы устройством очистки барабана *5*.

Однако в настоящее время более широкое распространение получил способ формирования скрытого изображения на поверхности барабана с помощью газоразрядной лампы (рис. 8.7, *б*). Свет от такой газоразрядной лампы *1* расщепляется на отдельные лучи с помощью системы оптоволоконных световодов *2*, и каждый луч модулируется индивидуальным ЖК-затвором. Совокупность таких затворов выполнена в виде единой панели *3*. Затем оптическая фокусирующая матрица *4* формирует отдельные пучки света, образуя горизонтальные ряды точек строки изображения. Развертка по вертикали создается вращением барабана *5*. Все остальные действия по переносу изображения на бумагу, закреплению изображения и подготовке поверхности барабана для печати следующей страницы выполняются так же, как описано выше. Для получения цветного отпечатка процедура печати выполняется трижды; при этом используются три красителя красного, зеленого и синего цветов. В последних моделях цветных лазерных принтеров все три красителя наносят за один оборот барабана, что значительно повышает скорость цветной печати.

Струйные принтеры обладают значительно меньшим быстродействием, но качество цветной печати у них очень высокое. Так, разрешение современных струйных принтеров составляет порядка 4800×1200 точка/дюйм, а время получения одного отпечатка — около 30 с. Обычно такие принтеры позволяют печатать как на стандартных листах бумаги формата А4, так и на формате 10×15 см, характерном для фотоснимков. Высокое качество печати и такой формат привели к тому, что струйные принтеры часто называют *фотопринтерами*. Печать производится посредством специальных сопел, выбрасывающих капли красителя размером около 2...3 пикалитров. Число сопел может достигать 512 на каждый цвет и в конечном итоге определяет качество получаемого отпечатка. Выброс струи производится в момент, когда сопло *2* (рис. 8.8) находится напротив печатаемой точки изображения. Импульсное дав-

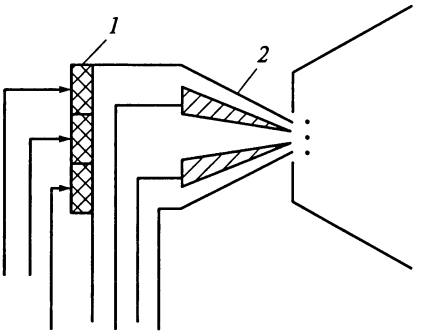


Рис. 8.8. Устройство сопла струйного принтера:

1 — пьезоэлектрический элемент; *2* — сопло

ление в сопле создается пьезоэлектрическим элементом *1*, а жидкий краситель может поступать из специального резервуара или представлять собой твердый краситель, размягчаемый под действием тока. Перемещение сопла или наличие нескольких сопел позволяет синтезировать контур в виде мозаики отдельных точек. Кроме того, наличие нескольких сопел дает возможность получать многоцветные изображения.

Струйный метод печати используют не только в принтерах, но и в специальных широкоформатных устройствах регистрации графической информации для получения различных плакатов, рекламы и т. п.

8.5. Накопители на магнитных дисках

Жесткие магнитные диски позволяют хранить огромные объемы информации и обеспечивают сравнительно быстрый доступ к ней. Однако для непосредственного использования хранимой на диске информации ее вначале необходимо переписать в оперативную память. Вся пользовательская информация на диске хранится в виде секторов, или блоков, размером по 512 или 2048 байт. Чтобы переписать нужную информацию с диска в ОЗУ, находят соответствующий сектор на диске, а затем организуют его чтение.

К характеристикам НМЖД относятся:

емкость, т. е. количество хранимой информации (емкость современных накопителей составляет 40... 180 Гбайт);

пропускная способностью или скорость записи и считывания;

время доступа, т. е. интервал времени от момента запроса до момента выдачи запрашиваемой информации.

Накопители на жестких магнитных дисках называют также накопителями с прямым доступом. Время на поиск нужного сектора определяется длительностью перемещения магнитной головки (МГ) на нужную дорожку и скоростью вращения диска.

Одной из важнейших характеристик НЖМД, обычно скрытых от пользователя, является *информационная плотность записи*, под которой понимают число бит, записанных на единице поверхности диска. Различают продольную плотность, т. е. число бит, записанных на одном миллиметре длины диска вдоль вектора скоро-

сти, и поперечную, т.е. число информационных дорожек на одном миллиметре вдоль радиуса диска. Плотность записи определяет размеры накопителя, его быстродействие и объемы его памяти. Плотность записи, в свою очередь, зависит от принципов регистрации информации, а также от материалов, конструкции и технологии изготовления диска и головки.

Принцип магнитной записи. Слой магнитного носителя, в котором хранится информация, выполняется из магнитотвердого материала с большими значениями коэрцитивной силы и остаточной индукции. Запись и считывание информации осуществляются посредством МГ, т.е. электромагнита, располагаемого над поверхностью носителя. Магнитная головка выполняется из магнитомягкого материала, чтобы при отсутствии тока в ней она не влияла на магнитный носитель.

Материал магнитного покрытия можно представить в виде хаотически расположенных доменов, ориентация которых изменяется под действием создаваемого головкой внешнего магнитного поля. При записи протекающий по обмотке МГ ток создает магнитный поток, который приводит к определенной ориентации магнитных доменов, оказавшихся под МГ. Домены ориентируются в одном из двух направлений в зависимости от направления тока в головке.

В настоящее время используют горизонтальную и вертикальную запись: при наиболее распространенной горизонтальной записи ориентация доменов производится в плоскости носителя (влево и вправо), при вертикальной — перпендикулярно плоскости носителя (вверх или вниз). Для реализации вертикальной записи необходимы более сложные МГ и конструкция самого носителя.

Изменение направления тока приводит к изменению направления ориентации доменов. При чтении головка позволяет определить моменты времени, когда под ней при движении носителя оказываются границы между участками носителя с противоположными состояниями намагниченности. Наводимая в головке ЭДС пропорциональна скорости изменения потока:

$$U = -k\mathrm{d}\Phi/\mathrm{d}t,$$

т.е. появление в обмотке головки импульса напряжения происходит тогда, когда под ней перемещается участок носителя с границей между состояниями намагниченности. Эту границу называют также отпечатком или переходом.

Соответствие отпечатков и записанных значений «0» и «1» называют *способом записи*. Он должен обеспечивать высокую плотность, помехоустойчивость и достоверность, а также возможность построения трактов записи и воспроизведения. Поскольку носитель может находиться всего в одном из двух состояний (он на-

магничен в ту или другую сторону), а для записи двоичного числа необходимо три состояния («0», «1» и пробел), то переходы не могут прямо соответствовать нулю и единице исходной последовательности (рис. 8.9, *а*).

Существует несколько способов записи: без возвращения к нулю, фазовой и частотной (рис. 8.9, *б*, *в*) модуляции, группового кодирования и ряд других. Но в настоящее время для записи информации на ЖМД чаще всего используют метод RLL 2,7, а для записи на гибкие магнитные диски (ГМД) — частотной модуляции.

Аппаратные средства, позволяющие записывать на носитель в соответствии с выбранным способом записи или восстанавливать двоичную информацию, называют *трактом*, или *каналом*, записи и воспроизведения. Основными компонентами тракта записи являются головки, усилители, детекторы информационных и син-

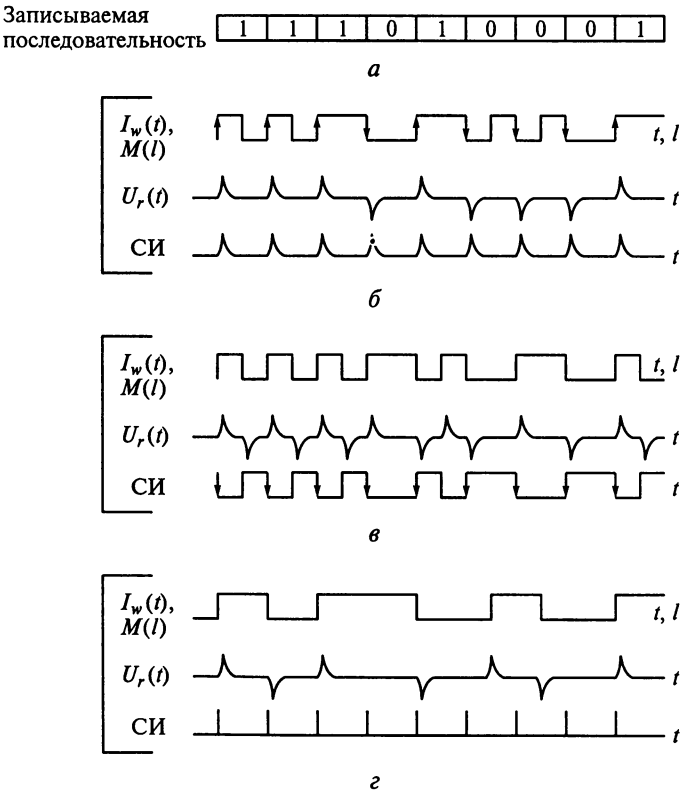


Рис. 8.9. Запись и воспроизведение информации на магнитном носителе:
а — исходная последовательность; *б* — фазовая модуляция; *в* — частная модуляция; *г* — модифицированная фазовая модуляция

хронизирующих сигналов и схемы управления записью и воспроизведением.

Частотная модуляция (ЧМ) представляет собой разновидность записи с самосинхронизацией. Запись «1» выполняется изменением направления тока внутри тактового промежутка, а начало каждого тактового промежутка отмечается сменой направления намагниченности. При записи «0» направление тока записи и, следовательно, состояние носителя не меняются. Сигнал, считываемый МГ при смене направления намагниченности, служит для синхронизации. Таким образом, при записи единиц частота переключения потока удваивается, а при записи нулей остается неизменной. Пример регистрации на носителе числа 111010001 приведен на рис. 8.9, *в*.

При чтении записанной информации границы тактовых промежутков восстанавливаются. Если в рамках восстановленного такта с МГ чтения получен импульс, то это соответствует «1», а если импульс в этом такте отсутствует — «0». При записи ЧМ необходима предварительная настройка логических схем тракта воспроизведения.

При записи способом ЧМ требуется довольно высокая стабильность скорости перемещения носителя, так как соответствующие единицам отпечатки должны располагаться строго посередине такового интервала. Кроме того, поскольку в каждом такте записи остается один (при записи нуля) или два (при записи единицы) отпечатка, то плотность записи не может быть очень большой. Для записи на ЖМД используют другие способы, требующие, однако, более сложных схем записи и воспроизведения.

Одним из таких способов является способ модифицированной фазовой модуляции (МФМ). Процесс записи и воспроизведения иллюстрируется на рис. 8.9, *г*. При записи «1» направление тока в головке записи меняется на противоположное в моменты, соответствующие началам тактовых интервалов T. Запись «0» производится изменением тока в головке записи в моменты, соответствующие серединам тактовых интервалов, т.е. направление изменения намагниченности значения не имеет. Таким образом, при записи числовой последовательности остаются отпечатки на расстоянии T, $1{,}5T$ и $0{,}5T$.

Чтобы избежать расположения отпечатков на расстоянии $0{,}5T$ друг от друга, переключение тока МГ при записи «0», непосредственно предшествующего «1», не производится. Тогда расстояния между отпечатками на носителе соответствует трем интервалам: T, $1{,}5T$ и $2T$.

При воспроизведении, если тракт «настроен» на начало тактового интервала и выделен интервал T (обнаружен импульс в начале такта), текущему биту приписывается значение «1». Если выделен интервал $1{,}5T$, то текущий бит равен «0», а схемы тракта

перестраиваются на середину такового интервала; если $2T$, то формируется сразу последовательность из двух бит «01». Если схемы «настроены» на середину такта, то интервал T соответствует «0»; если выделен интервал в $1{,}5T$, то формируется значение сразу двух бит «01», а схемы перестраиваются на начало тактового интервала.

Способ МФМ, иногда называемый групповым кодом 1,3, позволяет записывать информацию с более высокой плотностью, а следовательно, более экономно использовать дисковую поверхность. При высокой стабильности скорости перемещения носителя можно еще больше увеличить плотность, если не оставлять отпечатки нескольких следующих один за другим нулей. При воспроизведении число нулей определяется длительностью интервала между отпечатками, которые соответствуют единицам. Так формируется код RLL 2,7.

Для повышения достоверности данных записываемая на носитель информация кодируется с помощью корректирующего кода. Затем этот код записывается на носитель. При воспроизведении вначале восстанавливается корректирующий код и лишь затем исходный.

Структура накопителя на жестких дисках. В настоящее время наиболее распространенным типом является НЖМД с подвижными головками. На каждой поверхности располагается одна (или несколько) подвижных МГ. Перемещение блока МГ с дорожки на дорожку (а дорожки имеют вид концентрических окружностей) производится шаговым двигателем. Магнитные дорожки на разных поверхностях ЖМД, записанные при неподвижном состоянии блока МГ, образуют *цилиндр*. Информация записывается на дорожку блоками, причем для сокращения времени после заполнения одной дорожки производится запись на следующую дорожку того же цилиндра. (Таким образом удается избежать лишних перемещений головки.) Размер каждого информационного блока составляет 512 байт или 2 Кбайт. Он помещается в *сектор данных*, начинающийся с маркера и заканчивающийся контрольным кодом CRC, позволяющим исправлять однократные ошибки и обнаруживать многократные. На дорожках располагаются резервные секторы, которые служат для записи данных в случае обнаружения сбойных участков. После записи информации в резервный сектор, он становится основным, а сбойный помечается как дефектный и в него запись не производится. Для увеличения емкости накопителя на ЖМД число секторов на дорожках, находящихся на разных расстояниях от центра диска, переменно. При постоянной плотности размещения информации число секторов на внешних дорожках больше, на внутренних — меньше. Структура секторов на ЖМД в настоящее время не определена и может отличаться от структуры секторов, принятой в ГМД.

В герметичном корпусе НЖМД располагается двигатель, вращающий шпиндель с магнитными дисками (рис. 8.10). Головки (минимум по одной на каждой поверхности диска) располагаются на каретке, которая перемещается вторым двигателем ЛДВ; как правило, этот двигатель шаговый. На БУ этим двигателем поступает сигнал о текущем $A_т$ и конечном $A_к$ адресах. Чтобы конструкция получилась компактной, в современных накопителях головки перемещаются по дуге, а не по радиусу. Во время работы МГ «плывут» над поверхностью диска, не касаясь запоминающего слоя. За счет этого можно значительно увеличить скорость вращения дисков (а значит и скорость записи-считывания) до 7200 об/мин и выше, а также продлить срок службы накопителя до нескольких лет.

Поскольку каждый НЖМД содержит, помимо дисков, блок МГ и отдельные блоки приводов шпинделя и МГ (с усилителем мощности УМ, усилителем У и усилитеме сервомимтемы УСС), то совместимость таких накопителей обеспечивается на уровне электрических сигналов, что упрощает пользование такими накопителями.

Структура и особенности накопителя на гибких магнитных дисках. Конструкция накопителей на сменных ГМД сохраняется практически неизменной в течение 20 лет. Накопитель на ГМД в качестве запоминающей среды использует пластмассовый диск, покрытый тонким магнитным слоем. Запись и считывание информации выполняется контактным способом: МГ касается поверхности диска.

Неизменность конструкции обеспечивает совместимость со старым ПО и переносимость данных с компьютера на компьютер с помощью дискеты размером 3,5 дюйма. Дискеты в $5^1/_4$ дюйма больше не используются. Емкость дискеты измеряют числом байт пользователя (форматированная емкость) или общим числом байт (неформатированная емкость).

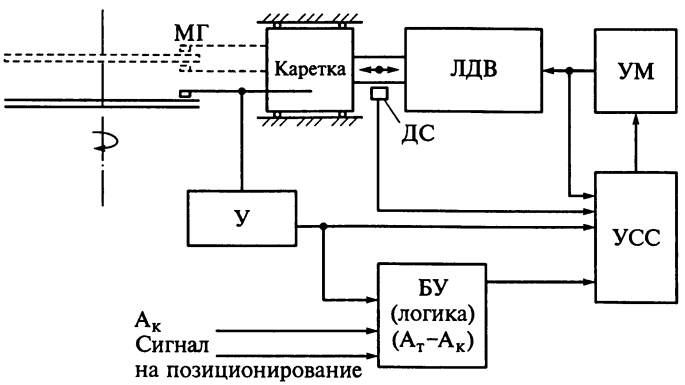


Рис. 8.10. Структура НЖМД

Дисководы могут иметь одну или две головки; первые позволяют считывать информацию только с одной стороны дискеты, вторые — с обеих сторон. Поэтому дискета может иметь форматированную емкость в 720 Кбайт или 1,44 Мбайт, неформатированная емкость таких дискет составляет 1 или 2 Мбайт. Одно время выпускались дисководы, предназначенные для дискет с форматированной емкостью в 2,88 Мбайт, но они не получили широкого распространения. Для того чтобы дисковод мог распознать, на какую емкость рассчитана дискета, в ее корпусе делают специальное отверстие (если емкость дискеты составляет 720 Кбайт, то отверстие отсутствует). В настоящее время в употреблении остаются дискеты объемом 1,44 Мбайт.

Информация размещается на дорожках, представляющих собой концентрические окружности. Скорость вращения дискеты довольно низкая (порядка 400 об/мин), а скорость записи или чтения информации не превышает 60 Кбайт/с.

В дисководе размещают два двигателя: один служит для вращения дискеты, а второй для перемещения головок. Обычно головки перемещаются с дорожки на дорожку с помощью шагового двигателя. Величина шага постоянна и строго фиксирована, она соответствует расстоянию между дорожками. Начало дорожки отмечается прямоугольным отверстием в центральной части дискеты, называемым *индексным отверстием*. Кроме того, в корпусе дискеты предусматривают отверстие для защиты от записи: если это отверстие закрыто, то запись разрешена, а если открыто — запрещена.

Дорожка разделяется на секторы фиксированного размера. В секторе может быть записано 512 или 2048 байт, причем в последнем случае каждый символ кодируется двумя байтами. Чтобы произвести запись или чтение, нужно указать адрес сектора, т.е. номер цилиндра, дорожки и сектора. Вначале необходимо переместить головки на нужный цилиндр, затем включить одну из двух головок и после этого найти сектор путем последовательного считывания информации всех секторов на дорожке.

Накопители на ГМД в настоящее время используются исключительно в качестве устройств ввода-вывода информации. Записанную информацию на одном компьютере можно прочесть на другом, благодаря стандартным размерам ГМД и стандартному представлению информации на дискете.

Формат дорожки ГМД представляет собой следующее. В начале дорожки, которое отмечается поступлением сигнала от датчика индекса, записывается промежуток G_1, содержащий 80 байт с кодом 4E. Затем записывается заголовок дорожки, состоящий из трех полей: SYNC (12 байт с кодом 00 для синхронизации контроллера), IAM (3 байт с кодом C2 и 1 байт с кодом FC) и промежуток G_2 (50 байт с кодом 4E). Вслед за заголовком дорожки размещается первый физический сектор, состоящий из 574 байт.

Он состоит из трех полей: идентификатора сектора (22 байта), промежутка идентификатора G_3 (22 байта с кодом 4E) и блока данных размером 530 байт (один байт маркера сектора, 512 байт данных и контрольный код CRC). Идентификатор сектора, в свою очередь, состоит из полей синхронизации (12 байт с кодом 00), четырехбайтовой метки заголовка (три байта с кодом A1 и одного байта FE), а также адреса сектора. Адрес сектора — это байт номера цилиндра, байт номера головки, байт логического номера сектора, байт размера цилиндра и два байта символа CRC. При форматировании, чтобы предусмотреть место для будущих символов, поле данных произвольно заполняется какими-либо байтами, например байтом символа NUL.

8.6. Накопители на компакт-дисках (CD-ROM, CD-R, CD-RW, DVD)

С середины 1980-х гг. получили распространение системы внешней памяти на компакт-дисках. В настоящее время такие диски широко используются для распространения ПО, баз данных, технических руководств, справочников и т.д. *Компакт-диск*, или CD (compact-disk), представляет собой пластмассовый диск с односторонней записью информации; обычно он бывает покрыт тонким отражающим слоем, например из алюминия. Этот слой и является запоминающей средой; цифровая информация заносится в нее в виде микроскопических углублений. Существует несколько типов компакт-дисков: допускающие только чтение (CD-ROM), однократную (CD-R) и многократную (CD-RW) записи. В последнее время получает распространение еще один вид оптических дисков, а именно DVD-диски.

CD-ROM. Компакт-диски, допускающие только чтение, выпускаются различного размера и емкости. Однако их диаметр не должен превышать 120 мм; диски такого размера могут быть установлены в устройство чтения на персональном компьютере. Информация записывается в виде секторов на спиралевидной дорожке, которая обеспечивает возможность воспроизведения аудио- и видеозаписей без наличия специальных буферных устройств, но затрудняет поиск данных, когда они хранятся в виде отдельных порций. В стандартном компакт-диске длина одной спиралевидной дорожки составляет 5,27 км, а при постоянной линейной скорости в 1,2 м/с чтение всей размещенной на дорожке информации произойдет за 73,2 мин. Расстояние между витками дорожки составляет 1,6 мк. Компакт-диски этого типа характеризуются сравнительно невысокой скоростью передачи информации (176,4 Кбайт/с), большим временем доступа и достаточно большой емкостью информации (650 Мбайт).

Обычно CD-ROM изготовляют методом «печати», т.е. переноса информации с мастер-диска. Вначале информацию записывают на мастер-диск с помощью сфокусированного лазера достаточно большой мощности. Он «прожигает» углубления, которые и переносятся на компакт-диск. Затем компакт-диск покрывают прозрачным лаком, защищающим его от пыли и царапин.

Считывание информации с диска выполняется посредством маломощного лазера. Луч этого лазера направлен на записанную дорожку и освещает вращающийся диск. Интенсивность отраженного от поверхности диска луча меняется в зависимости от того, попадает ли он на углубление или нет. Отраженный луч фиксируется фотодетектором, который преобразует изменение интенсивности луча в цифровые сигналы.

Чтобы обеспечить постоянство скорости считывания информации при постоянной угловой скорости вращения диска (CAV), на дорожках, находящихся на разных расстояниях от центра диска, углубления должны располагаться с различной плотностью: на внешних дорожках реже, а на внутренних чаще. Это приводит к тому, что внешние дорожки используются нерационально, поэтому такой метод не находит широкого применения. Вместо этого информацию размещают на диске в секторах одинакового размера, но чтение ее производят с постоянной скоростью. Для этого диск вращается с переменной скоростью, зависящей от положения лазерного луча для считывания информации. Этот метод получил название чтения с постоянной линейной скоростью (CLV). Угловая скорость вращения диска меньше, когда считывание информации производится с внешней дорожки диска, и больше по мере приближения луча к внутренней дорожке.

Данные на CD-ROM записываются в виде блоков (рис. 8.11). Каждый блок включает в себя поле синхронизации, состоящее из 12 байт, четырехбайтовое поле идентификатора, поле данных (2048 байт) и поле с корректирующим кодом (288 байт).

Поле синхронизации отмечает начало блока; оно состоит из байта, содержащего нули, десяти байт с единицами и двенадцатого байта со всеми нулями. Поле идентификатора содержит временную метку, адрес блока и режим. Нулевой байт режима указывает на пустое поле данных, режим 1 говорит об использовании корректирующего кода, а режим 2 — об отсутствии корректирующего кода и расширении поля данных до 2336 байт. Поля данных и корректирующего кода не требуют пояснений.

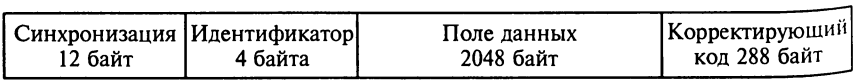


Рис. 8.11. Формат записи на CD-ROM

Головка при чтении и последовательном поиске блока должна находиться на дорожке и перемещаться перпендикулярно ей при прямом доступе. Однако алгоритм определения местонахождения блока достаточно сложен, что существенно замедляет его поиск.

CD-R. Оптические диски с однократной записью и многократным чтением CD-R служат для сохранения больших объемов часто используемой информации. Типичными областями применения таких дисков могут быть системы проектирования, бухгалтерский учет, резервное копирование и прочие системы архивного хранения документов.

Запись информации на CD-R осуществляется лазерным лучом относительно большой мощности. Пользователь с помощью достаточно мощного лазера форматирует диск в специальном накопителе, создавая на поверхности диска дорожку из последовательных пузырьков. Для записи информации отформатированный пузырьками диск помещают в накопитель, где посредством маломощного лазера можно разрушить («взорвать») пузырек. При чтении лазерный луч освещает дорожку, позволяя распознать наличие или отсутствие «взорванных» пузырьков, так как такой пузырек обладает более высоким контрастом.

Диски CD-R можно использовать для сохранения обновляемых файлов, однако в них не производится физического стирания старой записи и размещения на ее месте новой. Вместо этого при обновлении файла производятся запись на свободном месте диска под тем же именем и изменения в директории. К имени файла в директории дописываются специальные разряды, указывающие на то, что данная строка устарела, и создается новая строка с тем же именем файла. Поскольку на таком диске сохраняются все последовательные модификации файла, то становится возможным проследить все вносимые изменения.

CD-RW. По своему назначению оптические диски со стиранием информации CD-RW ближе всего соответствуют магнитным. Но они позволяют удалять информацию (вместе с покрытием), тем самым обеспечивая секретность. Кроме того, они оказались очень удобными для персональных компьютеров, обеспечивая переносимость информации между различными машинами. Среди многих предложенных технологий наиболее приемлемой оказалась магнитооптическая. Для записи и стирания информации в накопителях на дисках CD-RW используется энергия лазерного луча совместно с действием магнитного поля. Запись и стирание бита информации производятся лучом лазера, который локально нагревает отпечаток, при этом нагретый участок материала намагничивается в направлении внешнего магнитного поля, создаваемого катушкой (рис. 8.12, *а*).

При изменении направления намагниченности происходит поворот плоскости поляризации и соответственно изменяется ко-

эффициент отражения данного участка. В процессе чтения нужно определить направление магнитного поля по поляризации лазерного луча (рис. 8.12, *б*). Поляризованный свет, отраженный от участка, изменяет угол отражения в зависимости от направления намагниченности и принимается фотоприемником ФД. Процессы записи бита и его стирания отличаются только направлением тока в катушке подмагничивания.

Таким образом, операция записи на диске CD-RW включает в себя три цикла:

1) стирание всех битов на выбранном участке диска для записи новой информации;

2) запись новой информации во время следующего оборота диска;

3) контрольное считывание вновь записанной информации во время третьего оборота.

Следовательно, операция записи выполняется в три раза медленнее по сравнению с магнитными дисками (при одинаковой скорости вращения диска и той же плотности записи). Операция чтения требует всего один цикл. Современные магнитооптические покрытия допускают до 10^4 циклов перемагничивания; это наиболее критично для области диска, где размещается директорий.

DVD. Структура данных на DVD-диске зависит от его типа (DVD-ROM, DVD-R, DVD-RAM или DVD+RW). На DVD-дисках информация записывается на одной (односторонний носитель) или обеих сторонах (двухсторонний носитель) диска, причем диски DVD-ROM могут иметь с каждой стороны по одному или по два информационных слоя. Каждый информационный слой имеет спиралевидную дорожку с входной областью, областью данных и выходной областью. В области данных размещаются блоки данных пользователя, содержащие по 16 секторов. Размер физического сектора составляет 37 856 байт.

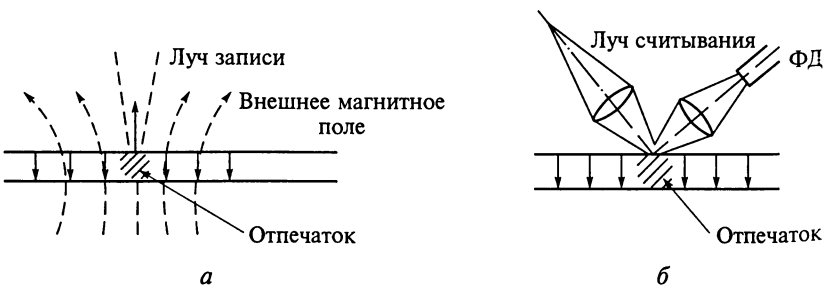


Рис. 8.12. Магнитооптическая технология в накопителе CD-RW:
а — запись; *б* — стирание

Номер физического сектора расположен в его заголовке. Адресация секторов по физическим адресам используется контроллером дисковода только для внутренних целей. Процессор, к которому подключен DVD-диск, обращается к нему по логическому адресу. *Логический адрес* — это адрес логического блока, размер которого составляет 2048 байт.

Обычно в дисководе находится лишь один лазер, а смена рабочей стороны диска выполняется вручную, поэтому методы адресации рассчитаны на работу с одним или двумя слоями, расположенными с одной стороны диска. На двухслойном диске адресация может выполняться как с параллельными, так и встречными путями треков.

При встречных путях треков область данных на первом слое диска заканчивается не выходной, а срединной областью, а второй слой начинается со срединной области. При переходе к другому слою дисковод меняет направление вращения диска на противоположное; при этом меняется фокусировка лазерного луча. Он фокусируется на втором слое. Физическая и логическая организация области данных на других типах DVD-дисков отличается от описанной, но незначительно.

8.7. Другие виды периферийных устройств

Помимо рассмотренных выше ПУ, как правило, входящих в стандартную комплектацию персонального компьютера, существует большое число разнообразных устройств, предназначенных для решения специфических задач, например стримеры, флэш-память, модемы, сканеры, графопостроители, трекболы и др. Все эти устройства служат для хранения и передачи информации между компьютерами или для ввода-вывода, т.е. преобразования форм представления информации, ее регистрации и отображения. Многочисленность используемых форм представления информации и методов их преобразования не позволяет в одном учебном пособии подробно рассмотреть каждую из них. Но некоторые ПУ, наиболее распространенные в настоящее время, заслуживают краткого описания.

Стримеры. Накопители на жестких магнитных дисках подвержены отказам и сбоям, поэтому существует вероятность потери всей хранящейся в них информации. Для ее сохранения нужно воспользоваться принципом резервирования, т.е. сохранения ее на нескольких носителях. Однако если для этого использовать второй НЖМД, то такое решение довольно дорого и не всегда решает поставленную задачу. (Подробно об использовании нескольких накопителей RAID-массива для обеспечения надежности хранения информации рассказано в гл. 12.) Поэтому во многих случаях

создают резервные копии информации на магнитной ленте с потоковой, или серпантинной, записью, т.е. с помощью стримеров.

При потоковой организации запись ведется последовательно по одной дорожке, хотя на ленте их несколько. После заполнения одной дорожки ленты запись продолжается на следующей дорожке. Для этого производится переключение головок записи (предусматривается по одной головке на каждую дорожку) и изменяется направление движения ленты на противоположное.

Запись на стример производится из НЖМД. Но для организации потоковой записи необходимо, чтобы поток записываемой информации был непрерывным, поэтому в стримере предусматривают буферную память, вмещающую несколько блоков. Однако если такая память все же окажется пустой, то потоковый режим записи будет нарушен. Восстанавливается он следующим образом. Пустое состояние буферной памяти фиксируется контроллером, и, если запись блока не завершена, вырабатывается сигнал, приводящий вначале к останову, а затем к движению ленты в противоположном направлении. При движении ленты в обратном направлении будет обнаружено начало последнего записанного на нее блока (возможно, не полностью). В этот момент вырабатывается сигнал, останавливающий лентопротяжный механизм и приводящий к повторному реверсированию направления движения ленты, т.е. она начинает перемещаться в первоначальном направлении. Первый выделенный сигнал начала блока при перемещении ленты в прямом направлении будет соответствовать блоку, запись которого не была завершена. Он будет использован для повторения операции записи из буферного ЗУ. Однако теперь это буферное ЗУ скорее всего уже будет заполнено, так как реверсирование движения ленты требует значительного времени, и запись блока завершится успешно.

Современные стримеры в большинстве случаев позволяют записывать на одну ленту всю информацию с одного НЖМД. Но время записи измеряется часами, поэтому операция копирования на стример производится в свободное время.

Флэш-память. В современных персональных компьютерах в качестве постоянной памяти для хранения BIOS (Basic Input-Output System) используют флэш-память, представляющую собой особый вид электрически стираемого программируемого ПЗУ. Впервые флэш-память появилась в конце 1980-х гг. и сразу завоевала широкое признание. Для хранения одного бита информации в ней используется всего лишь один транзистор, благодаря чему достигается высокая плотность расположения информации. Стирание информации в ней производится электрическими сигналами полностью или по блокам, а для записи информации не требуется специального программатора — она производится сигналами компьютера с помощью аппаратуры, реализованной в самой схеме

этой памяти. Содержимое флэш-памяти сохраняется без напряжения питания и не требует регенерации. Флэш-память выпускается в разных вариантах — с последовательным или параллельным интерфейсом, возможностью стирания всей информации из памяти, отдельных блоков разного или одинакового размера.

В последнее время флэш-память встраивается в цифровые фотокамеры, а для передачи снимков в компьютер она подключается к интерфейсу USB. Содержимое флэш-памяти может быть полностью стерто за одну или несколько секунд, а запись одного байта занимает около 10 мкс.

Модемы. С развитием компьютерных сетей надежная передача информации на значительные расстояния по каналам связи стала одной из важнейших задач. В качестве каналов связи обычно используются телефонные, радиорелейные, радиоканалы и др. При этом линиями передачи служат проводные, кабельные, волоконно-оптические, инфракрасные радиолинии и т.п. Канал связи принято характеризовать *пропускной способностью*, т.е. числом двоичных единиц информации, которое можно передать по нему за единицу времени, и достоверностью передачи, характеризуемой вероятностью искажения единицы информации.

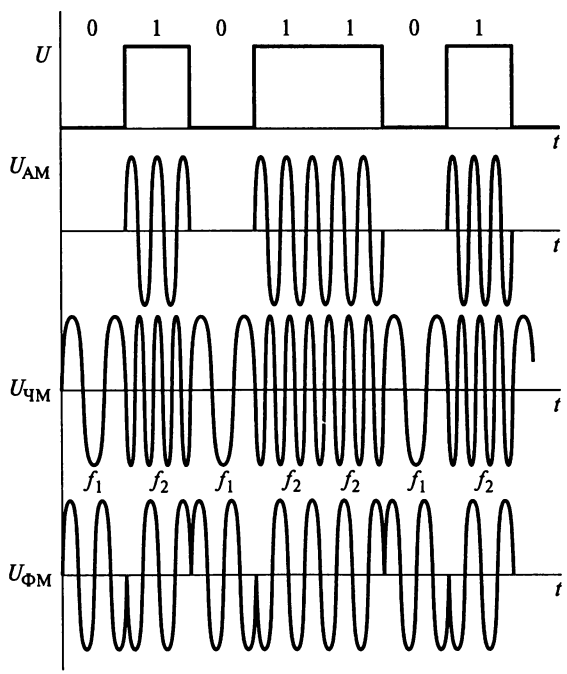


Рис. 8.13. Виды модуляции

Однако по существующим линиям связи нельзя передавать информацию в том виде, в котором она циркулирует в компьютере. Это вызвано тем, что при передачах на значительные расстояния линия связи начинает действовать в качестве фильтра с ограниченной частотой пропускания; кроме того, на передаваемую информацию оказывают влияние различные помехи. Все это приводит к недопустимо большим искажениям передаваемых прямоугольных импульсов.

Для ограничения полосы частот информационного сигнала можно осуществить модуляцию его сигналами несущей (высокой) частоты, которую и выполняет модем. На приемном конце модем должен выполнить обратное преобразование, т.е. демодуляцию сигнала. Известно несколько видов модуляции: амплитудная (АМ), фазовая (ФМ) и частотная (ЧМ). При амплитудной модуляции «1» передается наличием сигналов высокой частоты, а «0» — отсутствием таких сигналов (рис. 8.13). При фазовой модуляции при смене «1» на «0» и обратно производится изменение фазы сигнала на 180°, а при частотной модуляции «1» на «0» передаются сигналами разной частоты. Наиболее распространены фазовая и частотная модуляции. Это вызвано тем, что они обеспечивают наибольшую помехоустойчивость, хотя и требуют более сложного оборудования для своей реализации по сравнению с амплитудной.

Помимо преобразования сигналов для передачи по разным линиям на модемы возлагают и ряд других функций, например контроль, коррекцию ошибок, сжатие данных и т.д.

Контрольные вопросы

1. Какие особенности ПУ делают возможным организацию параллельной обработки и ввода-вывода?
2. Каковы основные функции системы ввода-вывода?
3. Что такое прямой доступ в память и как он реализуется? Какие устройства подключают к средствам прямого доступа?
4. Что такое программный ввод-вывод и как он происходит? Какими особенностями отличаются ПУ, работающие в режиме программного ввода-вывода?
5. Как организован ввод-вывод в компьютерах различных классов?
6. Как формируется код символа в клавиатуре? Как он передается в память компьютера?
7. Каково назначение мыши? Поясните принцип работы оптико-механической мыши.
8. На чем основана работа оптической мыши? Поясните принцип ее работы.
9. Каково назначение алфавитно-цифрового дисплея? Что такое маркер и как им управляют?
10. В чем принцип работы дисплея на базе ЭЛТ? Что такое частота регенерации изображения и чему она равна в современных дисплеях?

11. Как работает ЖК-дисплей? Что такое разрешающая способность? Какой разрешающей способностью обладают современные дисплеи?

12. Какие принтеры в настоящее время наиболее распространены? Какая разрешающая способность для них характерна?

13. Как работает струйный принтер? Какими достоинствами и недостатками он обладает?

14. Как работает лазерный принтер? Перечислите все этапы формирования изображения.

15. Что представляет собой ГМД? Какой стандартный объем пользовательской информации он вмещает?

16. Как хранится информация на гибком диске? Что такое дорожка, сектор, цилиндр? Как отмечается начало дорожки?

17. Приведите примерные временные характеристики накопителя на ГМД.

18. Какие особенности характерны для современных накопителей на ЖМД? Какие объемы информации они способны хранить?

19. Какие способы магнитной записи применяются сегодня для НЖМД, НГМД?

20. Приведите структурную схему НЖМД. Как он устроен?

21. Каковы достоинства и недостатки оптических дисков CD-ROM, CD-R, CD-RW?

22. Как производится запись информации на диски CD-ROM, CD-R?

23. Каким образом осуществляются запись и чтение информации с дисков CD-RW?

24. Чем вызвано малое быстродействие дисков CD-RW?

25. Что представляет собой стример? Как производится запись информации на ленту стримера?

26. Каково назначение стримера? Какими характеристиками он обладает?

27. Что представляет собой флэш-память? Где и как она используется?

28. Что такое модем? Какие задачи он решает?

29. Для каких каналов передачи данных предназначены модемы?

ГЛАВА 9

ОРГАНИЗАЦИЯ МУЛЬТИПРОЦЕССОРНЫХ И МНОГОМАШИННЫХ СИСТЕМ

9.1. Общие сведения

Компьютеры создавались для облегчения и ускорения вычислений и обработки больших объемов данных. Это требовало от них высокого быстродействия и надежности. Но последовательность выполнения команд, положенная в основу архитектуры фон-Неймана, не позволяла добиться быстрого и надежного решения задач. На каждом этапе развития вычислительной техники строились вычислительные системы (ВС), обладающие наивысшими показателями быстродействия, производительности, а часто и надежности. Отличительной особенностью таких суперЭВМ стала параллельная обработка информации. Для ее реализации требовались ВС, состоящие из множества процессоров и средств связи между ними; так появились многомашинные и многопроцессорные комплексы, а впоследствии и компьютерные сети.

Объединение нескольких процессоров в единую ВС увеличивает ее стоимость за счет необходимых дополнительных средств и наличия общей памяти, но позволяет экономить на ПУ.

Любой параллельный алгоритм может быть преобразован к последовательному виду, поэтому последовательная форма представления алгоритма является универсальной. Реализация его сводится к поочередному выполнению операций в единственном устройстве — процессоре. Прямая реализация параллельного алгоритма требует одновременной работы нескольких процессоров с последующей пересылкой промежуточных результатов между ними. Параллельная ВС должна соответствовать структуре алгоритма; она не может быть универсальной.

Параллельная ВС состоит из n процессоров. Приобретая ее, покупатель надеется ускорить процесс вычислений в n раз. Однако такого ускорения он никогда не получит. Это связано с тем, что невозможно представить программу, равномерно распараллеленную по n процессорам.

Процесс вычислений любого алгоритма в параллельной ВС развертывается как в пространстве, так и во времени. Структура параллельной ВС точно соответствует структуре алгоритма: все независимые операторы выполняются параллельно (т.е. одновре-

менно во времени) на отдельных процессорных элементах. Число таких процессорных элементов должно быть не меньше максимального числа параллельных процессов в каждый момент времени. Каждая задача имеет последовательную часть, например программу запуска или управления операциями ввода-вывода, которая должна выполняться только одним из процессоров. Промежуточные результаты также должны пересылаться между процессорами, однако в данном случае не учитывается время на пересылку и управление взаимодействием процессов, хотя эти управляющие процессы носят сугубо последовательный характер.

Допустим, что доля затрат времени на выполнение зависимых последовательных процессов составляет s (т.е. доля времени на выполнение сугубо последовательного кода), а доля затрат на выполнение независимых процессов (т.е. доля времени на выполнение частей программы, которые могут выполняться параллельно) в последовательной машине — p. Тогда очевидно, что $s + p = 1$. Пусть при этих условиях длительность решения задачи в последовательной машине составляет T_s.

Для ускорения обработки в «идеальной» параллельной машине предусмотрим N процессорных элементов, на которые возложим решение всех независимых процессов. Тогда время реализации того же алгоритма в параллельной машине

$$T_p = (s + p/N)T_s.$$

Коэффициент ускорения, т.е. отношение времени решения задачи на последовательной машине ко времени решения той же задачи на параллельной, определяется из следующего выражения:

$$k = T_s/T_p = 1/(s + p/N),$$

а при увеличении числа независимых процессов он асимптотически приближается к величине

$$k = 1/s.$$

Это выражение носит название *закона Амдала* (Д.Амдал — один из разработчиков всемирно известной системы IBM 360) и характеризует предельные возможности ускорения обработки за счет организации параллельных вычислений. Поскольку в любой программе имеется значительная доля последовательных операций, например, связанных с вводом-выводом, управлением, переходами и т.д., то возможности параллельной обработки ограничены. Кроме того, при параллельной обработке возникает необходимость в дополнительных действиях, связанных с обменом данными между отдельными процессорами, распределением задач и т.п.

Однако на больших параллельных ВС решается не одна, а сразу несколько независимых задач. Обычно пользователь стремится не сокращать время решения на параллельной ВС, а усложнить зада-

чу для нахождения более точных решений. Как правило, такое усложнение связано с увеличением параллельной части программы. Все это позволяет утверждать, что реальное ускорение при практических выполняемых задачах значительно выше. Впервые это положение было сформулировано Д. Густафсоном в 1980-х гг.

9.2. Классификация систем с несколькими процессорами

Чтобы ориентироваться в многообразии современных ВС, необходима определенная система классификации. Классификация ВС может выполняться с различных точек зрения, например с позиций пользователя, по архитектурным или структурным признакам и т.д. Так, пользователя интересует круг решаемых задач, производительность ВС, наличие средств программирования и отладки, а также стоимость ВС. С этих позиций все системы можно подразделить на универсальные, проблемно-ориентированные и специализированные; одно- и мультипроцессорные; персональные и серверы.

Если ВС обеспечивает высокие показатели быстродействия на широком классе задач, то она универсальна и способна решать задачи других классов, но при этом ее производительность значительно падает. Добиться высоких показателей быстродействия для широкого класса задач сложно и дорого. Для снижения затрат строят специализированные системы, обладающие высокой производительностью при решении задач из узкого класса. Во многих случаях специализированные системы не могут решать задачи других классов и оправданы только в том случае, когда выполняемые ими задачи встречаются достаточно часто или отсутствуют другие пути их решения. Такие ВС предназначены для систем реального времени при управлении объектами и технологическими процессами.

Проблемно-ориентированные ВС занимают промежуточное положение между универсальными и специализированными системами. Наиболее известны системы, предназначенные для числовой обработки (суперЭВМ), процессоры быстрого преобразования Фурье, обработки речевых сигналов, машины логических выводов и др.

Классификация по числу процессоров на первый взгляд значительно проще: все компьютеры принято делить на однопроцессорные и мультипроцессорные (или многопроцессорные). Однако на современном этапе развития технологии практически любой процессор персонального компьютера содержит несколько средств обработки, т.е. состоит из нескольких процессорных элементов, а в серверах может использоваться несколько процессоров, кото-

рые, в свою очередь, включают в себя несколько процессорных элементов.

Наибольший интерес представляет классификация ВС по архитектурным и структурным критериям, предложенная М. Флинном в 1966 г. Она основана на понятии потока, т.е. последовательности команд или данных. Все ВС Флинн предложил подразделять на четыре класса: ОКОД, ОКМД, МКОД и МКМД.

ОКОД (одиночный поток команд и одиночный поток данных). К этой группе относятся классические последовательные компьютеры, в которых в каждый момент может подвергаться обработке лишь один элемент данных по одной команде. В таких компьютерах может быть реализована конвейеризация команд, сопроцессирование, совмещение обработки и ввода-вывода.

ОКМД (одиночный поток команд и множественный поток данных). Высокопроизводительные системы этой группы позволяют выполнять одну операцию сразу над несколькими данными, т.е. в этих компьютерах существует общее управление (один поток команд), обеспечивающее одновременную и синхронную обработку в разных процессорных элементах нескольких данных. К числу таких ВС относятся матричные системы, состоящие из множества процессоров, управление которыми осуществляет единое управляющее устройство. Все процессоры получают одну команду и выполняют ее над локальными данными. К этому классу иногда относят и векторно-конвейерные ВС, в которых обработка элементов данных происходит со смещением во времени.

МКОД (множественный поток команд и одиночный поток данных). К таким ВС следовало бы отнести гипотетические системы, состоящие из множества процессоров, обрабатывающих один элемент данных, но под управлением различных команд. К данному классу иногда относят конвейерные системы, но большинство исследователей считают его пустым.

МКМД (множественный поток команд и множественный поток данных). Системы этого класса обладают несколькими процессорами, работающими со своими потоками данных и по своим командам. Этот класс очень широк: он включает в себя классические мультипроцессорные системы, машины потока данных, нейрокомпьютеры и многие другие. К этому классу относят и ВС, состоящие из нескольких автономных систем ОКМД, так называемые системы МОКМД.

Такая классификация позволяет оценить принцип работы ВС, поэтому часто бывает достаточной. Но в ней последний класс явно перегружен. Поэтому для более точного определения особенностей вычислительных систем Флинн предложил ряд дополнительных критериев: степень крупности операций, способ обмена результатами обработки, способ управления и синхронизации.

Процессор может выполнять операции, обладающие различной мерой крупности — от отдельных аппаратно реализуемых операций до процедур и программ. Поэтому ВС может иметь мелко-, средне- и крупноблочную структуру.

Для организации вычислительного процесса чрезвычайно важен способ обмена результатами обработки в отдельных процессорах. Такой обмен может осуществляться словами или сообщениями. Для пословного обмена необходимо наличие глобальной (общей для всех процессоров) памяти, разделяющей общее адресное пространство, или специальной коммуникационной сети. Системы с пословным обменом называют *сильносвязанными*. В них должны быть предусмотрены средства синхронизации процессов и защиты памяти. Обычно в таких системах используют единую разделяемую во времени систему ввода-вывода. Обмен сообщениями осуществляется в *слабосвязанных* системах. Для таких систем характерно наличие локальной памяти в каждом процессоре, асинхронная обработка, значительные затраты времени на обмен. Как правило, в слабосвязанных системах процессоры обладают автономными средствами ввода-вывода. (К таким системам можно отнести локальные сети.)

Важным критерием для оценки ВС служит способ организации связей между процессорами и модулями памяти. Обычно используют один из следующих способов связи: через разделяемую магистраль (общую шину), с помощью коммутаторов, посредством многовходовой памяти.

Способы управления и синхронизации важны для организации вычислительного процесса. Все ВС делят на синхронные и асинхронные, с централизованным и распределенным управлением. Если в ВС предусмотрена единая глобальная система синхронизации, осуществляющая жестким образом синхронизацию процессов, происходящих в различных процессорах, то такую систему называют *синхронной*. Для *асинхронных* систем характерно распределенное управление, когда каждый процессор обладает собственной системой распределенного управления и синхронизации.

Наконец, можно построить ВС, в которой реализация конкретного алгоритма происходит за счет изменения ее структуры: если перестройка структуры ВС производится до начала выполнения алгоритма, то такую ВС относят к статически, а если такая перестройка происходит в процессе выполнения алгоритма — то к динамически перестраиваемым.

Совместно используемая и распределенная память. Высокопроизводительные системы, в которых предусмотрена только общая совместно используемая память, часто называют *системами с однородным доступом* к памяти. Поскольку в таких системах все процессоры соединяются с модулями памяти при помощи какого-

либо коммутатора или шины, то производительность такой ВС определяется пропускной способностью шины. Обычно такие ВС содержат от 4 до 8 процессоров.

Если каждый процессор имеет собственную локальную память (в дополнение к общей), то ВС называют *системой с неоднородным доступом*. Наличие локальных памятей позволяет уменьшить число обращений в общую глобальную память и сделать систему эффективной при числе процессоров 20...30.

В многомашинных ВС каждый процессор обладает собственной памятью, к которой и может адресоваться; это *системы с распределенной памятью*. Доступ к памяти другого процессора происходит путем обмена сообщениями с ним. В таких системах (например, локальной сети) каждый процессор может самостоятельно изменять содержимое памяти, не заботясь о согласовании ее с другими процессорами. Но обмен сообщениями вызывает дополнительные издержки: нужно сформировать и передать сообщение, а принимающий процессор должен его обработать и передать ответное сообщение с требуемой информацией.

Когерентность кэш-памяти. В ВС, состоящих из нескольких процессоров и обладающих глобальной памятью, для уменьшения числа обращений к ней (а следовательно, и задержек) все процессоры снабжают собственной локальной кэш-памятью. При этом каждый процессор способен модифицировать данные в индивидуальной кэш-памяти по своей программе. Затем модифицированные данные могут направляться назад в одну и ту же ячейку глобальной памяти. Таким образом, значение в этой ячейке будет зависеть от того, какой из процессоров последним произвел ее модификацию. В этом и состоит проблема когерентности кэш-памяти. При изменении элемента данных в своей кэш-памяти одним из процессоров необходимо произвести соответствующие изменения в кэш-памяти других процессоров и глобальной памяти системы, чтобы сохранить состоятельность данных.

Для решения этой проблемы используют записи с аннулированием и обновлением. Любая строка в кэш-памяти помечается тегом, состоящим из двух битов состояния: *I* (недействительное), *V* (достоверное), *R* (резервированное) и *D* (измененное). При *записи с аннулированием*, если какой-либо процессор изменяет содержимое совместно используемого блока в своей кэш-памяти, то все копии этого блока в кэш-памяти других процессоров помечаются как недостоверные, для чего бит *V* в них устанавливается в «0». При сквозной записи этот модифицированный блок записывается также в глобальную память. При попытках этих процессоров прочитать отмеченный блок данных из своей кэш-памяти произойдет кэш-промах, т.е. появится необходимость прочитать модифицированный блок из глобальной памяти. При использовании обратной записи нужно вначале переписать этот

блок в глобальную память из того модуля, где он был модифицирован.

При *записи с обновлением* любые изменения в записи, произведенные в локальной кэш-памяти одного из процессоров, немедленно дублируются в кэш-памяти и всех остальных. Для внесения изменений в каждую кэш-память необходимо передать модифицированный блок данных всем процессорам, что возможно далеко не при всякой топологии сети и часто требует значительных затрат времени.

Для поддержания когерентности кэш-памяти может быть использовано несколько механизмов: совместно используемая кэш-память, некэшируемые данные, широковещательная запись, а также протоколы наблюдения и на основе справочника.

При совместно используемой кэш-памяти и отсутствии локальной памяти решение проблемы когерентности тривиально, но такая организация ВС не позволяет добиться высокой производительности: совместная кэш-память удалена от процессоров, и обращение к ней требует выполнения арбитражных функций.

Второй механизм — это запрет кэширования данных, которые могут быть изменены. Для реализации такого механизма нужно в памяти хранить признак, указывающий на возможность кэширования слова или блока данных. Поскольку модификация команд в программах обычно не производится, то нужно помечать лишь кэшируемые данные. В любом случае это приводит к дополнительным затратам на программирование.

Механизм широковещательной записи связан с передачей запросов на запись всей кэш-памяти системы, что заставляет все контроллеры проверять наличие копий модифицируемого блока. Этот механизм связан с дополнительными затратами времени на передачу сообщений.

Наибольшее распространение получили следующие аппаратные механизмы, реализующие протокол когерентности кэш-памяти: протоколы наблюдения и на основе справочника. В *протоколах наблюдения* когерентность обеспечивается контроллерами кэш-памяти. Обычно такие протоколы используют в мультипроцессорных ВС с общей шиной. Кэш-память содержит служебную информацию о состоянии информационного блока, взятого из общей памяти. Контроллеры каждой кэш-памяти «наблюдают» за шиной с помощью специального блока слежения, определяя запросы, способные изменить состояние когерентности совместно используемых блоков данных. Если в кэш-памяти какого-либо процессора находится модифицируемый другим процессором блок, то его контроллер аннулирует или обновляет его.

Существует несколько различающихся протоколов: сквозной, обратной, однократной записей, а также ряд протоколов, получивших названия от систем, в которых они были реализованы.

Простейший протокол *сквозной* записи предполагает, что запись в локальную память какого-либо процессора сопровождается записью в глобальную память, а все остальные процессоры объявляют свои копии недействительными. Недостаток — большой трафик общей шины.

Протокол с *обратной* записью несколько изменяет процедуру записи в глобальную память. Запись в нее модифицированного блока производится в случае соблюдения одного из условий: при удалении модифицированного блока из локальной памяти и обращении другого процессора к своей копии этого блока. Этот протокол далек от идеального, так как процессоры не узнают об изменении блока до его записи в глобальную память. Избежать этого недостатка можно, если процессор, модифицирующий блок, получает исключительные права на него.

Протокол с *однократной* записью использует запись с аннулированием. Первая запись в любую кэш-память происходит по схеме сквозной записи, а все другие процессоры объявляют свои копии этого блока недействительными. Недостаток этого протокола — необходимость первоначальной записи в глобальную память, даже если этот блок не используется другими процессорами. Объявление блоков недействительными производится сообщениями, содержащими отмеченные выше теги.

Все остальные протоколы представляют собой модификации этих трех и разработаны для различных мультипроцессорных систем.

Протоколы на основе справочника предназначены для сложных ВС с глобальной памятью и развитой сетью соединений между процессорами. В таких системах применение протоколов наблюдения неэффективно. Для реализации протокола на основе справочника необходимо собрать воедино всю информацию о содержимом локальной кэш-памяти в одном месте, называемом справочником, например в глобальной памяти. При обращении какого-либо процессора к глобальной памяти ее центральный контроллер, выполняющий функции контроллера справочника, обнаруживает этот запрос. Обращение к справочнику происходит каждый раз, когда любой процессор изменяет свою копию совместно используемых данных. В этом случае информация справочника служит для аннулирования или обновления копий этих данных во всех остальных видах кэш-памяти. Это протокол полного справочника.

Поскольку вся информация о состоятельности кэш-памяти сосредоточена в одном месте, то протокол полного справочника довольно прост. Однако недостатком такого справочника служит его размер, который пропорционален общему объему памяти. Для больших систем, состоящих из тысяч процессоров, накладные расходы на организацию справочника становятся слишком большими.

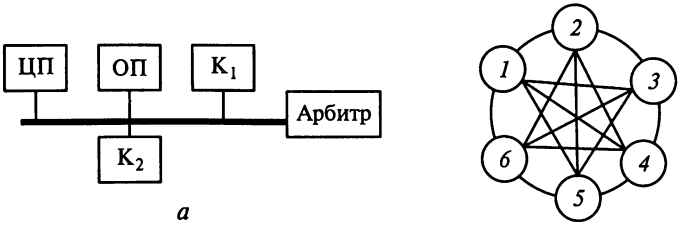
Уменьшить объем справочника можно, если организовать протокол ограниченного справочника или сцепленных справочников. В первом случае ограничивается число кэш-памяти, где могут находиться копии строки глобальной памяти. Во втором случае строится простейшая база данных, позволяющая вставлять указатели или удалять их.

Протоколы на основе справочника вызывают задержки из-за заторов в централизованном справочнике и задержек в коммуникационных сетях при обращении процессоров к глобальной памяти. Тем не менее они находят применение в сложных мультипроцессорных системах.

Подсистемы коммутации. Наличие нескольких процессоров в мультипроцессорной ВС еще не решает проблем повышения производительности и надежности. Необходимо все процессоры объединить некоторой коммуникационной сетью, служащей для обмена данными между ними. Такая коммуникационная сеть реализует физическое соединение отдельных узлов ВС, позволяя передавать данные между узлами, т.е. образуя единую вычислительную систему. В качестве узлов сети могут выступать процессоры, модули памяти, кластерные элементы и т.п. Различные узлы связываются между собой каналами передачи данных.

Требования к пропускной способности коммуникационной сети очень высоки, так как в значительной мере именно она определяет максимальную производительность системы в целом. Кроме того, сеть должна обеспечивать наращиваемость ВС. При этом нельзя забывать об экономических и технологических ограничениях.

Общая разделяемая шина представляет собой первый крайний случай коммуникационной сети; обмен данными между всеми процессорными элементами (ПЭ), включающими в себя ЦП, ОП и контроллеры $К_1$ и $К_2$, происходит с использованием этой шины (рис. 9.1, *а*). Однако, поскольку результаты в ПЭ формируются одновременно, а интенсивность обменов между отдельными ПЭ



а *б*

Рис. 9.1. Коммуникационная сеть:

а — общая разделяемая шина; *б* — полносвязанная коммуникационная сеть; *1...6* — процессорные элементы

во многих случаях очень высока, применение разделяемой шины может вызывать значительные потери производительности. Второй крайний случай — синхронная полносвязанная коммуникационная сеть; она связывает любой ПЭ с каждым из всех остальных отдельными каналами связи. Такая сеть, объединяющая n модулей, требует $n(n-1)$ каналов связи. На рис. 9.1, *б* такая сеть приведена для шести ПЭ. При большом числе процессорных элементов такая сеть становится нереализуемой.

Между этими сетями, представляющими два крайних случая, находится множество различных коммуникационных сред.

9.3. Конвейерные системы

При выполнении любой операции можно выделить ряд этапов: выборку и декодирование команды, подготовку параметров, выявление и разрешение конфликтных ситуаций, реализацию операции, запись результатов. Реализация операции может состоять из нескольких этапов, например при выполнении операции сложения с ПТ можно выделить этапы сравнения порядков, сдвига (выравнивания) порядков, сложения, округления и нормализации.

Если схему, реализующую данную операцию, разбить на несколько фаз, то на каждой фазе будет выполняться определенный этап этой операции. При обработке векторов $\overline{A}$, $\overline{B}$ этапы многократно повторяются (действительно, если вектор содержит n составляющих, то одна и та же фаза повторится n раз). Построим операционный конвейер, состоящий из нескольких последовательных схем ОУ, каждая из которых реализует один определенный этап операции. Между операционными схемами разместим регистры Рг для сохранения промежуточных результатов (рис. 9.2).

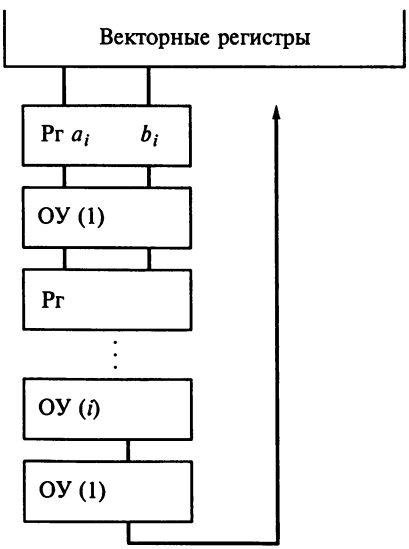


Рис. 9.2. Операционный конвейер

Между загрузками очередных векторов для обработки в операционный конвейер проходит интервал времени, называемый *тактом*. Величина такта определяется скоростью действий на каждой фазе конвейера, сложностью этих действий, наличи-

ем регистров между отдельными фазами и т.д. Обычно величина такта составляет $^1/_2 ... ^1/_8$ длительности машинного цикла, поэтому, если на выполнение какой-либо операции требуется четыре машинных цикла, то при наличии операционного конвейера производительность обработки повышается в 4...32 раза.

Загрузка операндов в операционный конвейер и запоминание результатов обычно производится в векторных регистрах или специально организованной локальной памяти. *Векторный регистр* — это совокупность обычных регистров, длина которых соответствует формату обрабатываемых данных, а их число — длине вектора. Выбор длины векторных регистров вызывает определенные трудности: если длина регистра меньше длины вектора, то это приводит к потере производительности, а если больше — к неэффективному использованию оборудования и снижению скорости работы конвейера.

Максимальное быстродействие ВС при выполнении векторных операций в конвейерном режиме

$$P_{max} = a/T_c,$$

где a — число конвейеров; T_c — длительность такта.

Дальнейшее повышение скорости обработки за счет увеличения числа ступеней, т.е. длины конвейера, не представляется возможным. Организация конвейера требует дополнительных схем, но и приводит к значительному повышению быстродействия. Конвейер не требует большого числа соединительных выводов, что важно при создании БИС. Кроме того, при работе конвейера производится не более трех обращений к локальной памяти на каждом такте. Все это привело к тому, что операционный конвейер стал неотъемлемой частью современных процессоров. Для эффективной обработки векторов различной длины можно организовать сцепление и распределение данных. Так, если две векторные операции следуют одна за другой, то результаты выполнения первой могут подаваться на вход второго конвейера без предварительного вычисления всех элементов вектора (рис. 9.3, *а*). Сцепление данных осуществляется при выполнении двух векторных операций:

$$\overline{C} = \overline{A} * \overline{B};$$
$$\overline{X} = \overline{C} + \overline{D}.$$

Распределение данных по нескольким операционным конвейерам (рис. 9.3, *б*) выполняется в случаях, когда векторные операции перемежаются со скалярными.

Однако эффективная производительность конвейерной системы ограничивается рядом факторов.

Максимальная производительность конвейерной системы достигается только при непрерывной загрузке конвейерного обра-

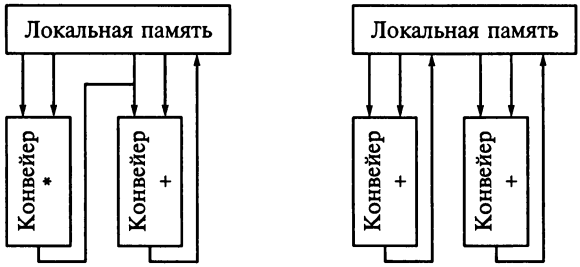


Рис. 9.3. Конвейерная обработка:
а — сцепление векторов; *б* — распределение

батывающего устройства, т.е. при бесконечной длине векторов, когда их элементы подаются на обработку непрерывно.

Реальные программы всегда содержат значительную долю скалярных операций.

На производительность процессора оказывают влияние не только операционные устройства, но и ОП, системы ввода-вывода и т.п.

Обычно возможности конвейерных систем принято характеризовать длиной вектора $L_{1/2}$, при которой производительность соответствует половине максимальной, или пиковой.

Широкое признание конвейерный принцип обработки получил с середины 1970-х гг., когда появилось несколько конвейерно-векторных ВС, наиболее известными из которых были Cray-1 и Cyber-205.

9.4. Симметричные системы

Среди MIMD-архитектур наибольший интерес сегодня представляют две. Такие архитектуры называют *симметричными* (Symmetric MultiProcessor — SMP) и *асимметричными* (Asymmetric MultiProcessor — AMP).

В MIMD-архитектурах несколько процессоров могут одновременно обращаться к программному ядру, что приводит к потенциальным задержкам и блокировкам. Одним из методов, позволяющим избежать блокировок, служит простой механизм: при обращении процесса к системной памяти осуществляется проверка замка. Если замок «заперт», то процесс ожидает его «отпирания». Этот метод обычно и называют асимметричным мультипроцессированием. Его преимущество заключается в простоте, а недостаток — в том, что процессорам часто приходится ждать, пока ядро не будет освобождено захватившим его процессором. Поскольку лишь один процессор может выполнять системную программу, то

эти замки устанавливают в начале каждого обращения к системной памяти и снимают их после его завершения. Вероятность ожидания возрастает с увеличением числа процессоров, и, следовательно, рост производительности за счет увеличения числа процессоров при асимметричной организации ограничен.

При симметричной организации все процессоры равноправны, связаны единой шиной и имеют доступ к общей памяти; по шине они также получают доступ к общим устройствам ввода-вывода (рис. 9.4). Все программы и данные хранятся в виде одной копии в единой памяти, к которой имеют доступ все процессоры и контроллеры ввода-вывода. Эта единая память позволяет ПО считать, что обработка производится как бы в одном процессоре, что значительно упрощает программирование. Наличие единой памяти снижает быстродействие, но память строят в виде нескольких банков: когда в одном банке происходит обновление информации, другой свободен для доступа. Кроме того, все процессоры снабжают кэш-памятью достаточно большого объема, служащей для устранения потерь времени из-за конфликтов при обращении к общей памяти и довольно медленной работы шины.

Наличие у каждого процессора своей кэш-памяти уменьшает потребность в обращениях к ОП, т. е. ускоряет обработку, но приводит к когерентности кэш-памяти.

Каждый из процессоров симметричной ВС выполняет задачи, возлагаемые на него ОП; поэтому эффективность SMP-системы зависит от способности ОС распределять задачи по процессорам. Для более эффективной работы ВС многие ОС (в частности, Windows NT) содержат средства разделения задачи на цепочки, или нити (thread), которые могут выполняться на разных процессорах. Это повышает не только производительность системы, но и ее надежность: выход из строя любого процессора может фиксироваться ОС, а его нагрузка автоматически распределяться по оставшимся исправными процессорам. Помимо высокой надежности симметричные системы обладают возможностью наращивать до определенных пределов свои вычислительные ресурсы, т. е. конфигурацию такой системы можно произвольным образом

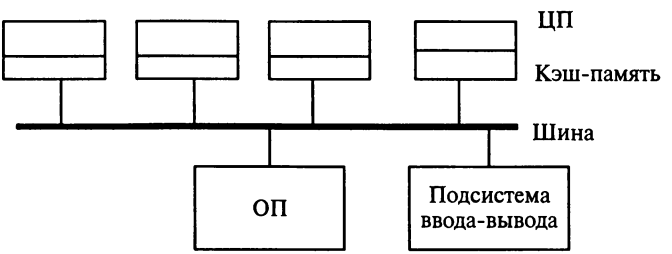


Рис. 9.4. Структура симметричной ВС

расширять. Система SMP обладает свойством масштабируемости и позволяет решить несколько задач: обеспечивает высокую производительность и надежность, обладает широкими возможностями наращивания вычислительных ресурсов и позволяет реализовать открытость информационной системы. Открытость системы обеспечивается стандартными средствами взаимодействия различных компонентов и частей ВС.

Примером симметричной ВС являются выпускаемые в настоящее время промышленные двухпроцессорные рабочие станции на базе процессора Pentium IV Xeon. Каждый процессор в этих станциях снабжен собственной кэш-памятью объемом не менее 512 Кбайт (обычно 1 Мбайт или больше). Помимо ЦП в этих станциях установлены графические платы, усовершенствованные графические порты и другие схемы, обеспечивающие преимущества при работе с трехмерной графикой, видеоклипами и т. п.

9.5. Системы со сверхдлинным командным словом

Анализ затрат времени на выполнение команды показывает, что требуется значительное время на обращение к ОП как за самой командой, так и за обрабатываемыми данными. Мультипроцессорная система не решает всех проблем, поскольку вся исходная, подлежащая обработке информация все равно находится в памяти, а доступ к ней ограничен. Единственный способ повышения производительности состоит в расширении доступа к памяти. Придание каждому процессору в мультипроцессорной системе «своего» блока памяти, построение ОП из нескольких блоков с поочередным обращением к ним, размещение в каждом процессоре собственной кэш-памяти составляют основу современных технических решений.

Концепция сверхдлинного командного слова (Very Long Instruction Word — VLIW) предполагает, что для загрузки нескольких обрабатывающих устройств можно получить необходимые команды и данные из памяти за одно обращение. Сверхдлинное командное слово позволяет разместить несколько независимых команд в одном слове и получать их из памяти за одно обращение. В машинах со сверхдлинным командным словом достигается заметная экономия времени при обращениях к памяти:

становятся доступными несколько задаваемых командным словом операций (нет необходимости несколько раз обращаться в ОП за новыми командами);

в регистрах процессора находится несколько результатов выполнения предыдущего командного слова. Эти результаты с большой вероятностью будут использованы в качестве входных операндов для следующего командного слова.

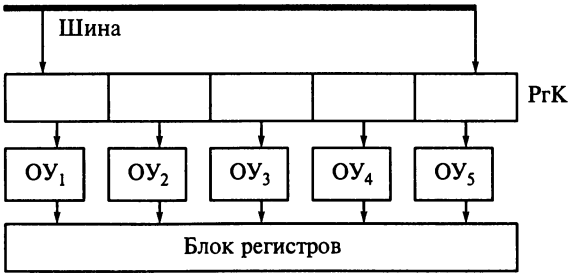


Рис. 9.5. Структура машины со сверхдлинным командным словом

Это создает предпосылки для уменьшения числа обращений к ОП, однако для того чтобы действительно уменьшить их количество, необходимо иметь широкую системную магистраль. Такая магистраль может быть реализована на печатной плате: современные магистрали имеют ширину 64, 128 и 256 разрядов. В машине со сверхдлинным командным словом (рис. 9.5) реализован синхронный принцип обработки, т.е. выборка следующего командного слова осуществляется по завершении строго определенного числа тактов. При реализации синхронной обработки важно, чтобы длительность всех операций была примерно одинаковой.

В качестве устройств обработки могут использоваться однотипные или разнотипные специализированные процессоры; для получения информации из памяти служат широкополосные магистрали. Принцип сверхдлинного командного слова в настоящее время находит широкое применение для построения мультипроцессорных серверов, так как не требует больших затрат на проектирование системы и создание ПО, но в то же время позволяет значительно повысить их быстродействие.

9.6. Другие виды мультипроцессорных систем

В настоящее время существует множество различных структур мультипроцессорных систем. Некоторые их этих структур находят применение в персональных компьютерах, некоторые служат основой для компьютеров сверхвысокого быстродействия, а какие-то пока изучаются на уровне моделей. Некоторые подобные структуры рассмотрены ниже.

Машины с массовым параллелизмом. Обычно машина с массовым параллелизмом (рис. 9.6) состоит из ведущего модуля (ВМ), модулей обработки (МО) и коммутирующей сети (КС), в них часто используется супервизорная магистраль (СМ).

В качестве ВМ используют ту или иную промышленную ЭВМ, которая служит для управления всеми МО, загрузки их данными

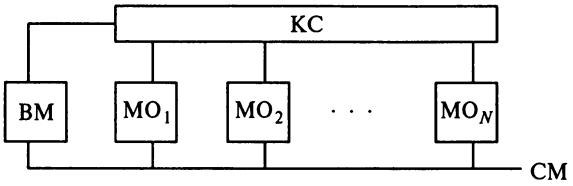


Рис. 9.6. Машина с массовым параллелизмом

и выдачи результатов. Коммутирующая сеть реализуется различными способами — от двухмерной до многомерной сети, дерева, тора и т. п. Супервизорная магистраль служит для передачи команд и приказов. Как правило, разрядность модуля обработки невелика — 1...16 бит. Частота синхронизации ограничена, тем не менее, она достигает 200...500 МГц.

К таким машинам предъявляются взаимоисключающие требования. Во-первых, число элементарных процессоров в них может достигать нескольких тысяч и, во-вторых, при таком большом числе ПЭ становятся значительно длиннее цепи управления и передачи данных, т. е. снижается тактовая частота.

Помимо МО общего назначения в состав таких машин могут входить различные специализированные модули. Примерами систем с массовым параллелизмом могут служить T3D фирмы Cray Research, Paragon фирмы Intel, CM5 фирмы Thinking Machines. Эти машины были разработаны в середине 1990-х гг. и позже.

Матричные системы. Высокопроизводительные системы такого типа представляют собой синхронные ОКМД-системы. В них используется большое число одинаковых обрабатывающих элементов, выполняющих одну операцию, но над различными данными (рис. 9.7, *а*). Большое число одинаковых СБИС делает их привле-

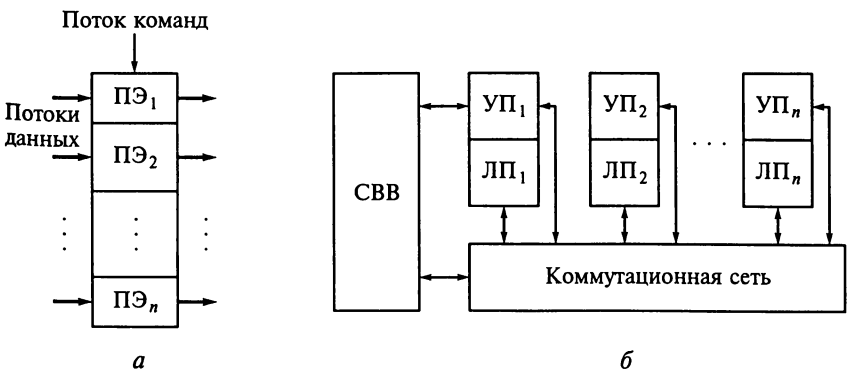


Рис. 9.7. Матричная ВС:

а — принцип матричной обработки; *б* — обобщенная структура

кательными не только с технологических позиций, но и с точки зрения повышения производительности. Так, если производительность одного ПЭ составляет $P_{ПЭ}$, то максимальная производительность всей ВС $P_{ВС}$ из n элементов будет в n раз выше:

$$P_{ВС} = nP_{ПЭ}.$$

Матричные системы предназначены для обработки векторов и матриц, когда однотипной обработке подвергается большое число элементов данных. Однако одновременный доступ в память для получения данных и сохранения результатов, а также необходимость выполнения операторов условной обработки приводит к ряду проблем и снижает производительность.

Управляющий процессор (УП) матричной ВС служит для передачи команд матричной обработки на обрабатывающие ПЭ, организует работу системы ввода-вывода данных и управляет КС. Каждый ПЭ обладает собственной локальной памятью (ЛП), в которой хранятся соответствующие элементы векторов или матриц (рис. 9.7, *б*). Обрабатывающие ПЭ предназначены только для выполнения арифметических и логических команд, но не команд условных переходов, которые реализуются в УП.

Процессорные элементы могут быть предназначены для обработки слов или отдельных разрядов; в первом случае матричные системы обычно служат для научно-технических расчетов и управления сложными процессами, а во втором — для обработки изображений.

Коммуникационные сети в матричных ВС должны обладать очень высокой пропускной способностью. В каждом ПЭ должен быть организован буфер команд достаточно большой глубины. Если отказаться от размещения операндов в общей памяти, то выборку команд и операндов можно выполнять параллельно. Все это приводит к тому, что эффективная производительность матричной ВС зависит не только от алгоритма, но и от размещения исходных данных.

Впервые идея матричной ВС нашла воплощение в проекте системы SOLOMON, но из-за конструктивно-технологических причин его реализация оказалась неэффективной. Дальнейшее развитие идея матричной обработки получила в системе ILLIAC-IV; был построен только один «квадрант» из 64 ПЭ, но он эксплуатировался для военных целей в США в течение нескольких лет начиная с 1974 г. В дальнейшем были созданы системы MasPar MP-1, Connection Machine, MPP, DAP-610 и ряд других. В них использовалась от 1024 до 65 536 одноразрядных и четырехразрядных (MP-1) ПЭ, а коммуникационная сеть строилась по принципу организации решетчатых связей с четырьмя ближайшими соседями, X-связей (MP-1) или по принципу гиперкуба (CM). Все эти матричные системы представляют собой архитектуры типа

ПЭ-ПЭ. Но в ряде матричных ВС коммуникационная сеть находится между ПЭ и модулями памяти; это системы типа ПЭ-память. К ним можно отнести BSP фирмы Burroughs и TRAC фирмы Texas Instruments.

Систолические системы. Проблема повышения производительности ВС связана с большими затратами времени при обращении в память. Для их снижения в современных ВС память организуют в виде нескольких уровней. Однако возможен и иной подход, совмещающий преимущества матричной и конвейерной обработки: все стадии обработки каждого элемента данных должны быть выполнены, прежде чем результат будет отправлен в память. Этот принцип реализуется систолической матрицей (рис. 9.8), в которой все ПЭ объединены прямыми и регулярными связями, образующими конвейеры. По этим конвейерам «прокачиваются» операнды, т.е. каждый элемент данных извлекается из памяти, ритмически продвигается по матрице ПЭ, и результат заносится опять в память.

При систолической обработке:

минимизируется число обращений в память, при обработке каждый элемент данных выбирается и заносится в память однократно крайними ПЭ;

облегчается решение проблем ввода-вывода вследствие уменьшения конфликтов при обращениях в память;

эффективно используются возможности СБИС за счет регулярности структуры систолической матрицы;

минимизируются длины связей между ПЭ за счет регулярности потоков данных и управляющих сигналов;

производительность систолической матрицы можно увеличить, добавляя в нее ПЭ.

Однако для реализации этих преимуществ необходимо найти соответствующие систолические алгоритмы, которые были созданы для широкого спектра задач. Большинство из них сводится к рекуррентным соотношениям того или иного вида.

Систолические матрицы могут обладать линейной, прямоугольной, гексагональной и трехмерной конфигурациями. Каждая конфигурация приспособлена для определенных функций: линейная — для реализации алгоритмов фильтрации при обработке сигналов и сравнения цепочек литер при обработке баз данных, прямоугольная — для перемножения матриц и нахождения двухмерного преобразования Фурье и т.д.

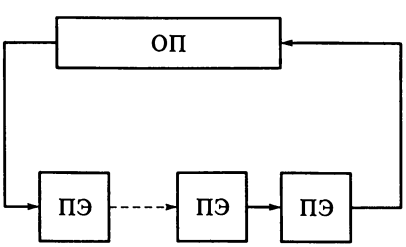
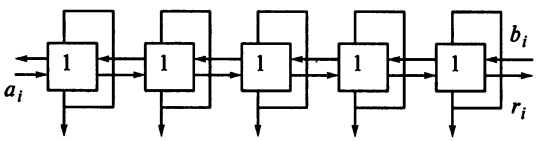


Рис. 9.8. Схема, характеризующая принцип систолической обработки

Для пояснения принципов действия систолической структуры рассмотрим реализацию операции поиска вхождений. *Поиск вхождений* — это поиск некоторой эталонной последовательности $B = (b_1, b_2, \ldots, b_m)$, где $b_j = 0$, 1 или (*) в исходной последовательности $A = (a_1, a_2, \ldots, a_n)$, где $a_i = 0$ или 1. Здесь символом (*) обозначено безразличное состояние соответствующего разряда.

Воспользуемся ПЭ, способными выполнять поразрядное сравнение и сохранять его результаты; объединим эти ПЭ в линейную структуру. На входы каждого ПЭ поступают значения a_i и b_j; каждый ПЭ производит сравнение и запоминает частичный результат r. Кроме того, он передает a_i и b_j на входы следующих ПЭ (рис. 9.9, *а*). Пусть производится поиск вхождений трехразрядного образца $B = (b_1, b_2, b_3)$ в последовательности $A = (a_1, a_2, \ldots, a_5)$. На рис. 9.9, *б* показана временна́я диаграмма выполнения операции сравнения цепочек литер. Вначале значения a_i и b_j загружаются в крайние ПЭ, откуда они начинают последовательно продвигаться по цепочке ПЭ. Элементы каждого потока, т.е. a_i и a_{i+1}, разделены периодом в один такт. На третьем такте в третий ПЭ попадут a_1 и b_1, где и произойдет их сравнение и будет выработан и сохранен



а

0	0	0	0	b_1
a_1	0	0	0	0
0	0	0	b_1	0
0	a_1	0	0	0
0	0	b_1	0	b_2
a_2	0	a_1	0	0
0	b_1	0	b_2	0
0	a_2	0	a_1	0
b_1	0	b_2	0	b_3
a_3	0	a_2	0	a_1
0	b_2	0	b_3	0
0	a_3	0	a_2	0
b_2	0	b_3	0	b_1
a_1	0	a_3	0	a_2

б

Рис. 9.9. Процесс сравнения цепочек литер:
а — линейная структура ПЭ; *б* — временна́я диаграмма

частичный результат сравнения r_{11}. Первый окончательный результат сравнения r_1 будет получен на седьмом такте в третьем ПЭ, второй результат сравнения — на восьмом такте во втором ПЭ и т.д.

Помимо ПЭ с «жесткими» связями существуют программируемые ПЭ. Однако увеличение набора операций ПЭ приводит к снижению производительности, усложнению соединений и значительным затратам времени на настройку. Кроме того, соединения в систолических матрицах имеют статический характер. Но указанные недостатки практически не проявляются в специализированных ВС.

Машины, управляемые потоком данных. В основе повышения производительности всех ВС лежит принцип совмещения операций и организации параллельной обработки. При этом в традиционных машинах команды выполняются в порядке их расположения в памяти, т.е. в ВС сохраняется последовательный характер командного управления, для чего используют счетчик (или в мультипроцессорных ВС несколько счетчиков) команд. Но наивысшей степени параллелизма можно достичь, если отказаться от дополнительных ограничений, присущих принципу командного управления. Альтернативными принципами является управление потоком данных и запросов.

При управлении потоком данных команда (инструкция) выполняется тогда, когда становятся доступными ее операнды, а при управлении потоком запросов — когда ее результат требуется другим командам.

В машинах, управляемых потоком данных (МПД), отсутствует понятие программы как последовательности команд, т.е. в ней нет счетчика команд. Команда передается на исполнение при создании условий для ее реализации. Одновременно при наличии достаточных аппаратных средств может выполняться произвольное число готовых к исполнению команд. Однако реализация принципа управления потоком данных вызывает ряд трудностей, к числу которых нужно отнести громоздкость программы, обработку итерационных циклов и работу со структурами данных.

Для описания обработки наиболее распространенной формой программы служит *граф потока данных* (ГПД). Он состоит из вершин (узлов) для обозначения необходимых операций и дуг (ребер), по которым передаются *метки*, или *токены*, *данных*. Точка вершины, в которую входит дуга, называется *входным портом*, или *входом*, а точка, из которой она выходит, — *выходом*. Срабатывание вершины означает выполнение инструкции; обычно оно происходит при наличии хотя бы одного токена на каждом из ее входов. Срабатывание вершины сопровождается удалением одного токена из каждого входного порта и размещением одного токена в выходном порту. На рис. 9.10 приведен пример простого ГПД для вычисления выражения $F = (x + y)^2$.

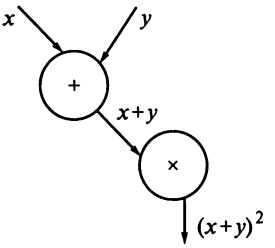


Рис. 9.10. Пример ГПД

Однако функциональные вершины позволяют строить лишь линейные программы, а в практических задачах возникает потребность в вычислениях при выполнении определенных условий. Для таких операций в ГПД используют вершины ветвления *BR* и слияния *M*. Вершина ветвления имеет два входных порта: *c* и *v*; на вход *v* подается токен данных, а на вход *c* — токен управления, в зависимости от значения которого токен данных передается на один из двух выходов вершины *t* или *f*:

если $c = 0$, то $v \rightarrow f$;
если $c = 1$, то $v \rightarrow t$.

Базовую архитектуру МПД можно представить в виде четырех устройств (рис. 9.11): устройства обработки (УО), состоящего из множества процессорных элементов; блока управления и памяти инструкций (ПИ), в каждой ячейке которого хранится инструкция и производится ее подготовка к исполнению; распределительной (РС) и селекторной (СС) сетей. Кроме этих устройств в системе должны быть предусмотрены средства ввода-вывода. Инструкция для двух операндов имеет следующий вид:

Op LO RO FL FR D,

где Op — код подлежащей выполнению операции; LO и RO — поля операндов; FL и FR — флаги готовности; D — поле назначения, определяющее инструкцию получателя результата.

Инструкция передается на выполнение по распределительной сети, когда в ней присутствуют два операнда (об этом говорят флаги готовности). Результат вновь передается в память по селек-

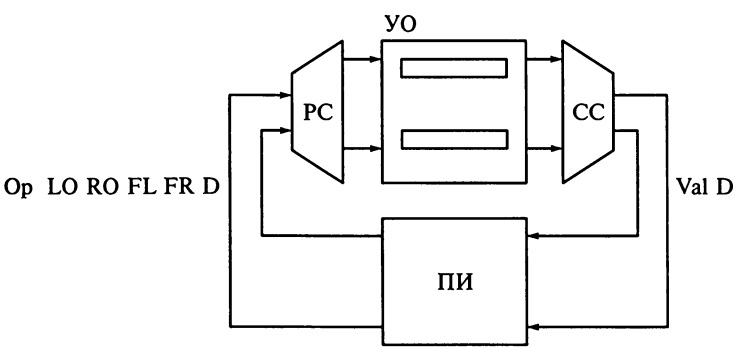


Рис. 9.11. Структура МПД

торной сети в виде значения Val и назначения D и попадает в соответствующую инструкцию. Если этот результат является вторым операндом, то новая инструкция вновь передается на исполнение; но если он служит лишь первым операндом, то происходит лишь установка соответствующего флага готовности операнда. Так продолжается до тех пор, пока не будет получен окончательный результат.

Рассмотренная МПД относится к статическим. Достоинство этой машины состоит в отсутствии дополнительных расходов времени на управление и синхронизацию исполнения инструкций; каждая инструкция выполняется, как только в нее поступают операнды. Однако система не обладает свойством повторной входимости, т.е. однократно использованный токен не может быть использован еще раз. Кроме того, каждый элемент данных должен рассматриваться как автономный объект; если же элементы объединить в структуру (например, список), то автономным объектом, в свою очередь, становится такая структура, т.е. очередная инструкция по ее обработке не может быть начата до завершения предыдущей.

Для построения динамических архитектур МПД, способных выполнять циклические вычисления, используют механизмы копирования и раскраски токенов. Механизм копирования предполагает возможность получения копий исполняемой программы. Из-за значительных затрат времени и объемов памяти при полном копировании предпочтение отдают организации с частным копированием, одним из вариантов которого является раскраска токенов. При раскраске каждому токену приписывается тег, определяющий номер итерационного цикла. Тогда поступающие во входные порты токены имеют различные теги, но исполнение инструкции происходит только при наличии в этих портах токенов с одинаковыми тегами. Динамические архитектуры значительно сложнее, однако они позволяют добиться более высокого уровня параллелизма при выполнении циклов и обрабатывать структуры данных.

В настоящее время промышленных МПД общего назначения не существует, однако принцип управления потоком данных находит применение в специализированных процессорах для обработки сигналов и изображений, а также для организации суперскалярности в процессорах Intel.

Машины, управляемые потоком запросов. Механизм управления по запросу состоит в исполнении инструкции, только когда для продолжения вычислений требуется ее результат. В основе такой модели лежит представление вычислительного процесса в виде графа, а обработка его вершин снизу вверх получила название *редукции графа*, поэтому машины, управляемые потоком запросов, называют также *редукционными*.

Для того чтобы вычислить выражение $a = (b + c)^2 - d/c$, необходимо спуститься вниз, породив запрос значений $(b + c)^2$ и d/c. Затем нужно найти значения $(b + c)$ и d/c и лишь после этого подняться на один шаг, т.е. найти значение $(b + c)^2$. Затем можно выполнить внешнюю операцию вычитания. Процесс порождения запросов продолжается до тех пор, пока не будет встречен оператор, готовый к выполнению. Этот процесс сопровождается ростом числа параллельных процессов, но его легко контролировать, т.е. можно задержать формирование запроса. Каждый выполненный оператор заменяется результатом, передаваемым снизу вверх, давая возможность найти результат предыдущего оператора. Можно сохранить найденный результат, заменив им соответствующую операцию, тогда при последующих обращениях к этому оператору повторного вычисления не потребуется. Известны две модели редукционных систем: строчная и графовая, отличающиеся тем, что передается в запрос — вычисленные значения или указатели на места их хранения. В настоящее время редукционные машины существуют только в виде моделей.

Нейрокомпьютеры. Высокопроизводительные системы, построенные на принципах параллельной обработки информации в распределенных нейронных сетях, моделирующих работу человеческого мозга, называются *нейрокомпьютерами.* Существует множество моделей, имитирующих внешние проявления этой работы.

Базовым элементом нейронной сети является линейный пороговый элемент (ЛПЭ), называемый *искусственным нейроном.* Сигнал на выходе ЛПЭ определяется соотношением

$$y = f(w_i x_i),$$

где f — пороговая функция; w_i — определенное значение весового коэффициента; x_i — входные сигналы.

Множество ЛПЭ группируются в линейные матрицы, называемые уровнями или слоями, которые, в свою очередь, группируются в сети. Информация в сети может передаваться в направлении от предыдущего уровня к последующему (такая сеть называется сетью с прямыми связями) и в обоих направлениях (сеть с обратными связями).

Нейронные сети способны запоминать образцы и взаимосвязи между данными без программирования: они «обучаются» путем ввода объектов и сравнивают входные образцы с хранящимися в их памяти. Такая сеть анализирует и классифицирует хранящиеся образцы в сравнении с входными, формирует новые весовые коэффициенты и, таким образом, обеспечивает самообучение.

Нейронные сети или нейрокомпьютеры обычно реализуют в виде программных моделей, а для сокращения длительности выполнения этих программ применяют специализированные сопроцессоры.

9.7. Проблемно-ориентированные системы

К числу специализированных и проблемно-ориентированных принято относить системы, предназначенные для работы в самых разных сложных условиях, высокая производительность в которых достигается при решении только определенных задач. Специализированные системы часто выполняют встроенными; они предназначены для решения узкого класса специфических задач и, как правило, не могут использоваться для других целей без переделок. Проблемно-ориентированные системы способны выполнять задачи более широкого класса: они могут строиться на базе RISC-процессоров и персональных компьютеров, а их ориентация определяется специфическими программами.

Примерами специализированных систем являются бортовые вычислительные машины, применяемые в автомобилестроении и авиации для решения задач навигации, управления различными радиолокационными устройствами, контроля бортовых систем, автоматического управления движением и т. п. Такие системы работают в режиме реального масштаба времени, обладают высокой надежностью и содержат множество специализированных периферийных устройств. Системы должны многократно выполнять одни и те же программы. Чаще всего исходная информация для них бывает представлена в аналоговом виде; ее нужно преобразовать в цифровую форму, а после обработки — вновь преобразовать к аналоговому виду.

Вторым примером может служить система обработки графической информации при машинном проектировании, моделировании, в компьютерной мультипликации, на телевидении. Системы обработки графической информации строятся как проблемно-ориентированные. Такая система должна формировать произвольные двухмерные и трехмерные цветные изображения, выполнять операции масштабирования, переноса, формирования проекций, исключения невидимых фрагментов, осуществлять раскраску и выделение отдельных фрагментов.

Третьим примером являются системы для работы с базами данных (БД), призванные ускорить выполнение некоторых функций по управлению БД и повысить производительность ВС при решении информационно-логических задач и задач искусственного интеллекта. Эти системы могут быть как специализированными, так и проблемно-ориентированными.

Рассмотрим подробнее структуру сигнальных процессоров и пример построения анализатора спектра на базе двух таких процессоров. *Сигнальные процессоры*, т. е. процессоры, предназначенные для обработки аналоговых сигналов, выпускаются в виде БИС и могут содержать в себе аналого-цифровые и цифроаналоговые преобразователи (АЦП и ЦАП). Так, простейший микропроцес-

сор К1813 для обработки сигналов (рис. 9.12) содержит цифровую и аналоговую части. В цифровую часть входят: ПЗУ программ на 192 слова длиной в 24 разряда, двухпортовое статическое ОЗУ на 40 слов длиной в 25 разрядов, память констант на 16 слов, специальный регистр (СР), используемый для преобразования входных напряжений, и операционное устройство. Операционное устройство содержит 25-разрядное АЛУ и масштабирующее устройство (МУ). Кроме того, цифровая часть содержит устройство управления и синхронизации с обычным для компьютеров счетчиком команд и тактовым генератором.

В аналоговую часть этой БИС входят ЦАП, компаратор (К), а также два мультиплексора: входных ($М_{вх}$) и выходных ($М_{вых}$) сигналов со схемами выборки-хранения. Схема цифрового обмена (ЦСО) переводит работу аналоговых входов в режим последовательного ввода.

Система команд отражает специфику данного процессора, ориентированного на обработку аналоговых сигналов. Формат команды имеет длину 24 бита и разбит на пять полей:

КОпАЛУ[3], АдрА[6], АдрВ[6], Сдвиг[4], АнК[5].

Поле кода аналоговой операции АнК предназначено для управления аналоговой частью, а остальные четыре поля — цифровой. Масштабирующее устройство позволяет быстро за счет сдвига производить умножение цифровой информации на константу. Число операций невелико: восемь арифметико-логических, две команды перехода и возврата, а также пять команд условной арифметико-логической обработки. Условные команды в поле АнК содержат номер бита регистра СР, значение которого определяет условие выполнения цифровой команды.

В более сложной БИС процессора TMS320 аналоговых элементов нет. Преобразователи сигналов подключаются к этой БИС в качестве внешних устройств. В дополнение к внутренней памяти программ можно подключать дополнительную внешнюю память.

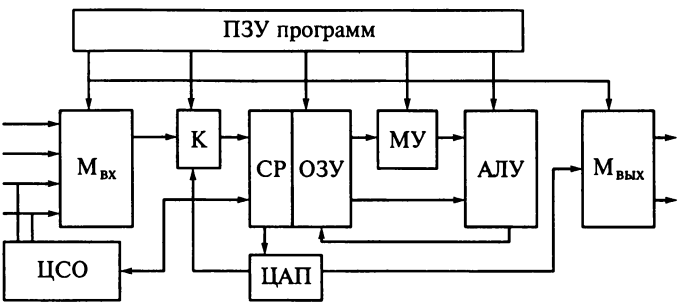


Рис. 9.12. Структура аналогового процессора К1813

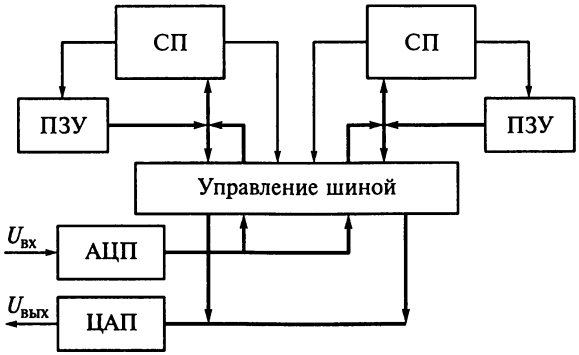


Рис. 9.13. Двухпроцессорный анализатор спектра

Операционная часть содержит 32-битное АЛУ с аккумулятором, сдвиговые регистры, матричный умножитель 16 × 16, а также внутреннюю память данных емкостью 144 × 16.

На базе такого процессора можно строить мультипроцессорные системы, в качестве примера которых приведем двухпроцессорный анализатор спектра (рис. 9.13). Каждый процессор использует внешнюю память программ.

Примером специальной ВС для обработки графических изображений является система «Титан», в состав которой входят четыре векторных процессора и дополнительные графические средства (рис. 9.14). Векторный процессор использует принцип конвейерной обработки и наделен большим регистровым файлом — аналоговым запоминающим устройством (АЗУ), объединенным с целочисленным процессором и процессором обработки элементов изображения. Собственно операции с дисплейным файлом осуществляются целочисленным процессором. Одна часть разделенной шины служит для загрузки векторного процессора, а вто-

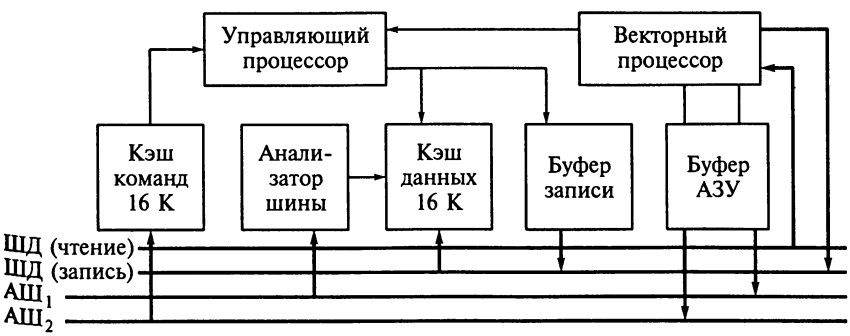


Рис. 9.14. Мультипроцессорная ВС «Титан»

рая — для обмена данными между целочисленным процессором, векторным процессором, памятью, графической подсистемой и портами ввода-вывода. Процессоры обладают кэш-памятью объемом 16 Кбайт.

Эта специализированная ВС позволяет строить трехмерные изображения со скоростью до 600 тыс. векторов/с.

Рассмотрим некоторые особенности *машин баз данных* (МБД), представляющих собой аппаратные средства, предназначенные для ускорения работы ВС с БД. В них реализованы архитектурные и структурные принципы параллельной обработки информации. Такие системы должны сохранять данные во внешней памяти в течение длительного времени, поэтому помимо средств ускорения доступа к этим данным должны быть предусмотрены меры, обеспечивающие нормальную работу ВС при сбоях или отказах дисковых накопителей (см. гл. 12).

Любое обращение к БД — это обращение к внешней памяти, требующее значительного времени. Поэтому вначале МБД представляли собой специализированные аппаратные средства, предназначенные для повышения производительности ВС при решении информационно-логических задач и уменьшения времени реакции системы при запросах к БД. Впоследствии на МБД стали возлагать и другие функции: управление запросами, распределение данных, выполнение реляционных отношений и т. п.

Машины баз данных принято характеризовать числом рабочих мест и затратами времени на обработку запросов к БД. Обычно МБД предназначаются для большого числа рабочих мест, а затраты времени в общем случае зависят от числа одновременно находящихся в системе запросов. Чаще всего МБД характеризуют временем реакции и пропускной способностью системы.

Время реакции системы — это интервал между моментом завершения ввода запроса пользователя к БД и началом поступления ответного сообщения. Этот интервал в основном определяется длительностью поиска информации в БД и временем ее обработки.

Пропускная способность МБД определяется числом обработанных запросов за единицу времени. Оба этих показателя зависят от текущей нагрузки на ВС, т. е. ее увеличение приводит к повышению пропускной способности, но и к ухудшению реакции системы.

В заключение рассмотрим возможную структуру МБД и приведем ряд примеров. На рис. 9.15 показана обобщенная структурная схема МБД. Запросы от пользователей поступают в блок трансляции запросов (БТЗ), где формируются последовательности внутренних команд МБД. Эти последовательности передаются на исполнение в ПЭ, ориентированные на выполнение реляционных операций. Команды, связанные с поиском информации на дис-

ках, передаются в контроллер ввода-вывода (КВВ), где и происходит ее поиск. Найденная информация Д передается через ОП в процессорные элементы для реализации запрошенных операций.

Для ускорения операций по ассоциативному поиску данных во внешней памяти и сокращения объемов передаваемой информации между ПЭ и накопителями на дисках служат фильтр-процессоры. Они обладают оперативной памятью значительного объема, где может разместиться информация из нескольких секторов. После завершения первого цикла фильтрации информация может быть вновь записана на диск для продолжения многокритериального поиска.

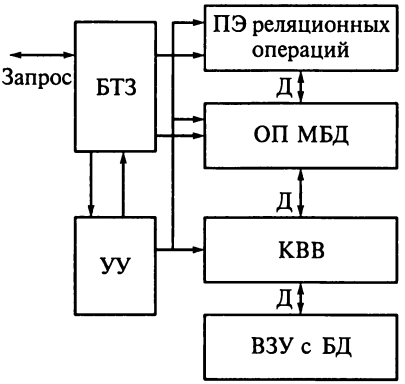


Рис. 9.15. Структурная схема МБД

В дальнейшем создавались многопроцессорные МБД, в которых огромные БД хранились на нескольких автономных дисковых накопителях. Нужно, однако, помнить, что характеристики таких МБД будут зависеть от распределения информации по накопителям. Для ускорения и повышения надежности таких машин хранящуюся на дисках информацию дублируют. К таким МБД относятся DBC/1012 фирмы Teradata, iPSC/II фирмы Intel и др.

Большинство современных МБД обладают рядом общих черт, несмотря на различие их структуры. Они поддерживают реляционную модель данных, подключаются к ведущему компьютеру в качестве ведомой машины и позволяют не только уменьшить время реакции на запрос к БД, но и повысить пропускную способность системы. Однако такие МБД целесообразны только при очень больших БД; кроме того, время реакции системы зависит от типа запроса и распределения данных по дисковым накопителям.

Контрольные вопросы

1. Как принято характеризовать современные мультипроцессорные ВС? Что представляет собой классификация Флинна?
2. Что такое система с конвейерной обработкой? Какие ВС выделяют в класс конвейерных?
3. Что представляет собой структура ВС SIMD-типа? Каким образом в ВС этого типа производится загрузка данных для обработки в процессорах?
4. Как выглядит структура ВС MIMD-типа? Каким образом происходит обмен информацией между отдельными процессорами?

5. Что представляют собой ВС, реализующие симметричный принцип обработки?

6. Какие аппаратные средства используются для организации симметричной обработки?

7. Что такое асимметричная ВС? Имеет ли такая система какие-либо преимущества перед симметричной?

8. Что представляет собой ВС со сверхдлинным командным словом? Как производится обработка информации в такой системе?

9. Какими преимуществами обладает конвейерная система и как она работает?

10. Какое максимальное ускорение обработки можно достичь, используя конвейерный принцип обработки? Чем ограничивается число ступеней в конвейере?

11. При каких условиях производительность конвейерной ВС будет максимальной? Чем она определяется?

12. Какие основные характеристики отличают машину с массовым параллелизмом?

13. Какие трудности возникают при попытках использования кэш-памяти в мультипроцессорной ВС? Как они разрешаются?

14. Сравните классические машины, управляемые потоком команд, с машинами потока данных.

15. Что лежит в основе машины потока данных? Что требуется для активации операции в статической МПД?

16. В чем принцип действия редукционной машины? Что сдерживает появление таких машин?

17. Что положено в основу нейрокомпьютера? Что представляет собой линейный пороговый элемент?

18. Какими преимуществами обладает систолический компьютер? В чем заключается принцип его действия?

19. Поясните принцип действия матричных систем. Приведите пример промышленных компьютеров матричного типа.

20. Какие ВС принято называть специализированными и проблемно-ориентированными?

21. С какой целью строят МБД и какие задачи на них возлагают?

ГЛАВА 10

ОРГАНИЗАЦИЯ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА

10.1. Общие сведения

Организация вычислительного процесса — это процесс формирования и управления вычислительной нагрузкой. Решение этой задачи возлагают на ОС, предназначенную для управления аппаратными, программными и информационными ресурсами компьютера. Такая ОС представляет собой комплекс программ, обеспечивающий контроль над существующими ресурсами компьютера, их распределением и использованием. Операционная система действует как интерфейс между приложениями и пользователями, с одной стороны, и аппаратными средствами, с другой. В этом смысле ОС решает две основные группы задач:

1) предоставление удобной для пользования расширенной виртуальной машины вместо реальной аппаратуры компьютера;

2) рациональное управление ресурсами компьютера и повышение эффективности их использования.

Операционная система как виртуальная машина. Для того чтобы успешно решать свои задачи, современный пользователь или даже прикладной программист может обойтись без досконального знания аппаратуры компьютера. Ему не обязательно быть в курсе того, как функционируют различные электронные блоки и электромеханические узлы компьютера. Более того, очень часто пользователь может не знать даже системы команд процессора. Он привык иметь дело с мощными высокоуровневыми функциями, которые ему предоставляет ОС.

Так, например, при работе с диском программисту, создающему приложение для работы под управлением ОС, или конечному пользователю ОС достаточно представлять его в виде некоторого набора файлов, каждый из которых имеет определенное имя.

Последовательность действий при работе с файлом заключается в его открытии, выполнении одной или нескольких операций чтения или записи, а затем в закрытии файла. Такие частности, как используемый метод записи на магнитный носитель или текущее состояние механизма перемещения МГ, не видны программисту. Именно ОС скрывает от программиста большую часть осо-

бенностей аппаратуры и предоставляет возможность простой и удобной работы с требуемыми файлами.

Если бы программист работал с аппаратурой компьютера напрямую без участия ОС, то для организации чтения блока данных с диска ему пришлось бы использовать более десятка команд с указанием множества параметров, например номера блока, номера сектора и т.д. После завершения операции обмена с диском он должен был бы предусмотреть в своей программе анализ ее результата. Учитывая, что контроллер диска способен распознавать более 20 различных вариантов завершения операции, можно считать программирование обмена с диском на уровне аппаратуры не самой тривиальной задачей. Не менее обременительной выглядит и работа пользователя, если ему для чтения файла с терминала потребовалось бы задавать числовые адреса дорожек и секторов.

Операционная система избавляет программистов не только от необходимости напрямую работать с аппаратурой НЖМД, предоставляя им простой интерфейс, но и осуществляет другие рутинные операции, связанные с управлением аппаратными устройствами компьютера: физической памятью, таймерами, принтерами и т.д.

В результате реальный компьютер, способный выполнять только небольшой набор элементарных действий, определяемых его системой команд, превращается в виртуальную машину, выполняющую широкий набор гораздо более мощных и разнообразных функций. Виртуальная машина тоже управляется командами, но командами другого, более высокого уровня, например удалить файл с определенным именем, запустить на выполнение некоторую прикладную программу, повысить приоритет задачи, вывести текст на печать. Таким образом, одно из назначений ОС — предоставление пользователю (программисту) некоторой расширенной виртуальной машины, которую легче программировать и с которой легче работать, чем непосредственно с аппаратурой реального компьютера.

Операционная система как система управления ресурсами. Операционная система не только предоставляет удобный интерфейс к аппаратным средствам компьютера, но и является механизмом, распределяющим ресурсы компьютера. К числу основных ресурсов современных ВС могут быть отнесены процессоры, оперативная память, таймеры, наборы данных, внешняя память, принтеры, сетевые устройства и др.

Управление ресурсами ВС с целью наиболее эффективного их использования и является назначением ОС. Например, мультипрограммная ОС организует одновременное выполнение сразу нескольких процессов на одном компьютере, поочередно переключая процессор с одного процесса на другой, исключая его простои, вызываемые обращениями к вводу-выводу.

Операционная система также отслеживает и разрешает конфликты, возникающие при обращении нескольких процессов к одному и тому же ресурсу. Критерий эффективности, в соответствии с которым ОС организует управление ресурсами компьютера, может быть различным. Например, в одних системах важна производительность вычислительной системы, в других — время ее реакции. Операционные системы организуют вычислительный процесс по-разному, соответственно выбранному критерию эффективности.

10.2. Базовые понятия и определения

Любая ОС оперирует некоторым набором базовых понятий, на основе которых строится логика ее функционирования. Например, подобными базовыми понятиями могут быть задача, задание, процесс, набор данных, файл, объект. Одним из наиболее распространенных базовых понятий ОС является процесс.

Процесс — это совокупность машинных команд и данных, исполняющаяся в рамках ВС и обладающая правами на владение некоторым набором ресурсов. Эти права могут быть эксклюзивными, когда ресурс принадлежит только этому процессу. Некоторые из ресурсов могут разделяться, т.е. одновременно принадлежать двум и более процессам, в этом случае говорят о *разделяемых ресурсах.*

Возможны два варианта выделения ресурсов процессу:

1) предварительная декларация использования тех или иных ресурсов (до начала выполнения процесса в систему передается перечень ресурсов, которые будут использованы процессом);

2) динамическое пополнение списка принадлежащих процессу ресурсов по ходу выполнения процесса при непосредственном обращении к ресурсу.

Реальная схема выделения ресурсов зависит от конкретной ОС. На практике возможно использование комбинации этих вариантов. Для простоты изложения будем считать, что в нашем случае ОС имеет возможность предварительной декларации ресурсов, которые будут использованы процессом. Необходимо отметить, что любая ОС должна обладать такими свойствами, как надежность, защита, эффективность и предсказуемость.

Ядро — это резидентная часть ОС. В ядре размещаются программы обработки прерываний и драйверы наиболее «ответственных» устройств. Это могут быть и физические, и виртуальные устройства. Например, в ядре могут располагаться драйверы файловой системы, ОЗУ. Следующие уровни структуры — динамически подгружаемые драйверы физических и виртуальных устройств. Это драйверы, добавление которых в систему возможно «на ходу» без

перекомпоновки программ ОС. Они могут являться резидентными и нерезидентными, а также работать как в режиме супервизора, так и в пользовательском режиме.

Выделяют следующие основные логические функции ОС: управление процессами, управление ОП, планирование, управление устройствами и файловой системой.

10.3. Управление процессами

Важнейшей частью ОС, непосредственно влияющей на функционирование ВС, является подсистема управления процессами. Приведем основные особенности этой подсистемы.

Жизненный цикл процесса. Рассмотрим типовые этапы обработки процесса в системе, совокупность этих этапов будем называть *жизненным циклом процесса* в системе. Он содержит следующие этапы:

образование (порождение) процесса;

обработка (выполнение) процесса;

ожидание (по тем или иным причинам) или постановка на выполнение;

завершение процесса.

Однако жизненные циклы процессов в реальных системах могут иметь свою, системно-ориентированную совокупность этапов.

Рассмотрим в качестве примера некоторую абстрактную ОС. Пусть имеется специальный *буфер ввода процессов* (БВП), т.е. пространство, в котором размещаются и хранятся сформированные процессы от момента их образования до момента начала выполнения. На данном этапе происходит формирование всех необходимых структур данных, соответствующих процессу. В частности, на этом этапе ОС формирует информацию о предварительно заказанных ресурсах данным процессом. Основная задача БВП — «подпитка» системы новыми процессами, готовыми к исполнению.

После начала выполнения процесс попадает в *буфер обрабатываемых процессов* (БОП), в котором размещаются все процессы, находящиеся в системе при мультипрограммной обработке. Обобщенный жизненный цикл процесса можно представить в этом случае *графом состояний*. Рассмотрим кратко переходы процесса из состояния в состояние (рис. 10.1).

0. После формирования процесс поступает в очередь на начало обработки в ЦП (попадает в БВП).

1. В БВП выбирается наиболее приоритетный процесс для начала обработки в ЦП (попадает в БОП).

2. Процесс прекращает обработку в ЦП по причине ожидания операции ввода-вывода, поступает в очередь завершения операции обмена (БОП).

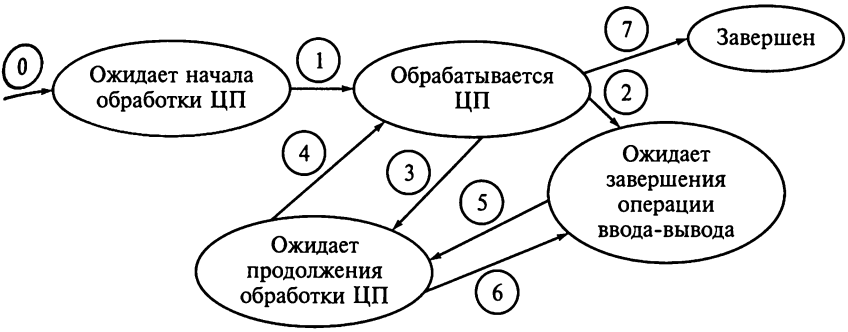


Рис. 10.1. Граф состояний процесса

3. Процесс прекращает обработку в ЦП, но в любой момент может быть продолжен (например, истек квант времени ЦП, выделенный процессу). Поступает в очередь процессов, ожидающих продолжения выполнения ЦП (БОП).

4. Наиболее приоритетный процесс продолжает выполнение в ЦП (БОП).

5. Операция обмена завершена, и процесс поступает в очередь ожидания продолжения выполнения в ЦП (БОП).

6. Переход из очереди готовых к продолжению процессов в очередь процессов, ожидающих завершения обмена (например, ОС откачала содержимое адресного пространства процесса из ОЗУ во внешнюю память) (БОП).

7. Завершение процесса, освобождение системных ресурсов. Корректное завершение работы процесса, разгрузка информационных буферов, освобождение ресурсов (например, реальный вывод информации на устройство печати).

Текущее состояние любого процесса из БОП изменяется во времени в зависимости от самого процесса и состояния ОС. С каждым из процессов из БОП система ассоциирует совокупность данных, характеризующих актуальное состояние процесса — *контекст процесса*. (В общем случае контекст процесса содержит информацию о текущем состоянии процесса, включая информацию о режимах работы процессора, содержимом регистровой памяти, используемой процессом, системной информации ОС, ассоциированной с данным процессом.)

Процессы, находящиеся в одном из состояний ожидания, в своих контекстах содержат всю информацию, необходимую для продолжения выполнения — состояние процесса в момент прерывания (копии регистров, режимы ОП, настройки аппарата виртуальной памяти и т.д.). Соответственно при смене выполняемого процесса ОС осуществляет «перенастройку» внутренних ресурсов ЦП, происходит смена контекстов выполняемых процессов.

Операционная система обеспечивает возможность корректного взаимодействия процессов — от передачи сигнальных воздействий от процесса к процессу до организации корректной работы с разделяемыми ресурсами.

Контекст процесса может состоять из следующих составляющих:

пользовательская — состояние программы как совокупности машинных команд и данных, размещенных в ОЗУ;

системная — содержимое регистров и режимов работы процессора, настройки аппарата защиты памяти, виртуальной памяти, принадлежащие процессу ресурсы (как физические, так и виртуальные).

Типы процессов. *Полновесные процессы* — это процессы, выполняющиеся внутри защищенных участков памяти ОС, т.е. имеющие собственные виртуальные адресные пространства для статических и динамических данных. В мультипрограммной среде управление такими процессами тесно связано с управлением и защитой памяти, поэтому переключение процессора с выполнения одного процесса на другой является достаточно дорогой операцией. В дальнейшем под процессом будем подразумевать полновесный процесс.

Легковесные процессы, называемые еще *нитями* или *сопрограммами*, не имеют собственных защищенных областей памяти. Они работают в мультипрограммном режиме одновременно с активировавшей их задачей и используют ее виртуальное адресное пространство, в котором им при создании выделяется участок памяти под динамические данные (программный стек), т.е. они могут обладать собственными локальными данными. Нить описывается как обычная функция, которая может использовать статические данные программы. Для одних ОС можно сказать, что нити являются некоторым аналогом процесса, а в других — представляют собой части процессов. Таким образом, в любой ОС понятие «процесс» включает в себя:

исполняемый программный код;

собственное адресное пространство, представляющее собой совокупность виртуальных адресов, которые может использовать процесс;

ресурсы системы, которые назначены процессу ОС;

хотя бы одну выполняемую нить.

При этом процесс может включать в себя понятие исполняемой нити, т.е. однонитевую организацию «один процесс — одна нить».

В данном случае понятие процесса связано с понятием отдельной и недоступной для других процессов виртуальной памяти. С другой стороны, в процессе может несколько нитей, т.е. процесс может представлять собой многонитевую организацию.

Нить также включает в себя понятие контекста — это информация, которая необходима ОС для того, чтобы продолжить выполнение прерванной нити. Контекст нити содержит текущее состояние регистров, стеков и индивидуальной области памяти, которая используется подсистемами и библиотеками. В данном случае характеристики нити во многом аналогичны характеристикам процесса. С точки зрения процесса, нить можно определить как независимый поток управления, выполняемый в контексте процесса. При этом каждая нить, в свою очередь, имеет свой собственный контекст.

Взаимодействие процессов. Методы синхронизации. Процессы, выполнение которых хотя бы частично перекрывается по времени, называются *параллельными*. Они могут быть независимыми и взаимодействующими. *Независимые* процессы используют независимое множество ресурсов, и на результат работы такого процесса не влияет работа независимого от него процесса. *Взаимодействующие процессы*, наоборот, совместно используют ресурсы, и выполнение одного может оказывать влияние на результат другого.

Совместное использование несколькими процессами ресурса ВС, когда каждый из процессов одновременно владеет ресурсом, называют *разделением ресурса*. Разделению подлежат как аппаратные, так программные ресурсы. Разделяемые ресурсы, которые должны быть доступны в текущий момент времени только одному процессу, — это так называемые *критические ресурсы*. Таковыми ресурсами могут быть, как внешнее устройство, так и некая переменная, значение которой может изменяться разными процессами.

Необходимо уметь решать две важнейшие задачи:

1) распределение ресурсов между процессами;

2) организация защиты адресного пространства и других ресурсов, выделенных определенному процессу, от неконтролируемого доступа со стороны других процессов.

Важнейшим требованием мультипрограммирования с точки зрения распределения ресурсов является следующее: результат выполнения процесса не должен зависеть от порядка переключения между выполняемыми процессами, т.е. от соотношения скорости выполнения процесса со скоростями выполнения других процессов.

Рассмотрим ситуацию, изображенную на рис. 10.2. Символ, считанный процессом *A*, был потерян, а считанный процессом *B* — выведен дважды. Результат выполнения процессов здесь зависит от того, в какой момент осуществляется переключение процессов и какой конкретно процесс будет выбран для выполнения следующим. Такие ситуации называют *гонками* между процессами, а процессы — *конкурирующими*. Единственный способ избежать го-

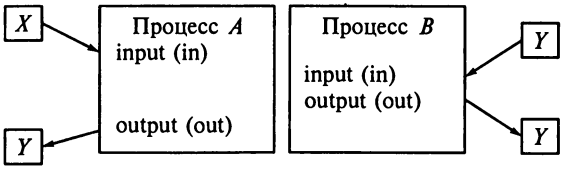


Рис. 10.2. Конкуренция процессов за ресурс

нок при использовании разделяемых ресурсов — контролировать доступ к любым разделяемым ресурсам в системе. При этом необходимо организовать *взаимное исключение*, т.е. такой способ работы с разделяемым ресурсом, при котором постулируется, что в тот момент, когда один из процессов работает с разделяемым ресурсом, все остальные процессы не могут иметь к нему доступ.

Проблему организации взаимного исключения можно сформулировать в более общем виде. Часть программы (фактически набор операций), в которой осуществляется работа с критическим ресурсом, называется *критической секцией*, или *критическим интервалом*. Задача взаимного исключения в этом случае сводится к тому, чтобы не допускать ситуации, когда два процесса одновременно находятся в критических секциях, связанных с одним и тем же ресурсом.

Организация взаимного исключения актуальна не только для взаимосвязанных процессов, совместно использующих определенные ресурсы для обмена информацией. Возможна ситуация, когда процессы, не подозревающие о существовании друг друга, используют глобальные ресурсы системы, такие как устройства ввода-вывода, принтеры и т.п. В этом случае имеет место конкуренция за ресурсы, доступ к которым также должен быть организован по принципу взаимного исключения.

При организации взаимного исключения могут возникнуть *тупики*, т.е. ситуации в которых конкурирующие за критический ресурс процессы взаимно блокируются. Есть два процесса A и B, каждому из которых в некоторый момент требуется иметь доступ к двум ресурсам R_1 и R_2. Процесс A получил доступ к ресурсу R_1, следовательно, никакой другой процесс не может иметь к нему доступ, пока процесс A не закончит с ним работать. Одновременно процесс B завладел ресурсом R_2. В этой ситуации каждый из процессов ожидает освобождения недостающего ресурса, но оба ресурса никогда не будут освобождены, и процессы никогда не смогут выполнить необходимые действия.

Существуют различные механизмы организации взаимного исключения для синхронизации доступа к разделяемым ресурсам. К числу классических методов относятся семафоры Дейкстры и мониторы Хоара.

Семафоры Дейкстры. *Семафор* представляет собой переменную целого типа S, над которой определены две операции: down(S) (или $P(S)$) и up(S) (или $V(S)$). Оригинальные обозначения P и V, данные Дейкстрой и получившие широкое распространение в литературе, являются сокращениями голландских слов proberen — проверить и verhogen — увеличить.

Операция down(S) проверяет значение семафора, и если оно больше нуля, то уменьшает его на 1. Если же это не так, процесс блокируется, причем операция down считается незавершенной. Важно отметить, что вся операция является неделимой, т.е. проверка значения, его уменьшение и, возможно, блокирование процесса производятся как одно атомарное действие, которое не может быть прервано. Операция up(S) увеличивает значение семафора на 1. При этом если в системе присутствуют процессы, блокированные ранее при выполнении down на этом семафоре, ОС разблокирует один из них с тем, чтобы он завершил выполнение операции down, т.е. вновь уменьшил значение семафора. При этом также постулируется, что увеличение значения семафора и, возможно, разблокирование одного из процессов и уменьшение значения являются атомарной неделимой операцией.

Чтобы прокомментировать работу семафора, рассмотрим пример. Представим себе супермаркет, посетители которого прежде чем войти в торговый зал должны обязательно взять себе инвентарную тележку. В момент открытия магазина на входе имеется N свободных тележек — это начальное значение семафора. Каждый посетитель забирает одну из тележек, уменьшая тем самым число оставшихся на 1, и проходит в торговый зал — это аналог операции down. При выходе посетитель возвращает тележку на место, увеличивая число тележек на 1, — это аналог операции up. Допустим, очередной посетитель обнаруживает, что свободных тележек нет — он вынужден «блокироваться» на входе в ожидании появления тележки. Когда один из посетителей, находящихся в торговом зале, покидает его, ожидающий тележку посетитель «разблокируется», забирает тележку и проходит в зал. Таким образом, семафор в виде тележек позволяет находиться в торговом зале (аналоге критической секции) не более чем N посетителям одновременно. Положив $N = 1$, получим реализацию взаимного исключения. Семафор, начальное (и максимальное) значение которого равно 1, называется *двоичным семафором* (так как имеет только два состояния: «0» и «1»).

Семафоры являются низкоуровневыми средствами синхронизации, для их корректной практической реализации необходимо наличие специальных, атомарных семафорных машинных команд.

Мониторы Хоара. Идея монитора впервые сформулирована Ч.Хоаром в 1974 г. В отличие от других средств, монитор представляет собой *языковую конструкцию*, т.е. некоторое средство, пре-

доставляемое языком программирования и поддерживаемое компилятором. *Монитор* — это совокупность процедур и структур данных, объединенных в программный модуль специального типа. Постулируются три основных свойства монитора:

структуры данных, входящие в монитор, могут быть доступны только для процедур, входящих в этот монитор (таким образом, монитор представляет собой некоторый аналог объекта в объектно-ориентированных языках и реализует инкапсуляцию данных);

процесс «входит» в монитор путем вызова одной из его процедур;

в любой момент времени внутри монитора может находиться не более одного процесса. Если процесс пытается попасть в монитор, в котором уже находится другой процесс, он блокируется. Таким образом, чтобы защитить разделяемые структуры данных, их достаточно поместить внутри монитора вместе с процедурами, представляющими критические секции для их обработки.

Процедуры и данные монитора имеют особую семантику, поэтому первое условие может проверяться еще на этапе компиляции, кроме того, код для процедур монитора тоже может генерироваться особым образом, чтобы удовлетворялось третье условие. Поскольку организация взаимного исключения в данном случае возлагается на компилятор, число программных ошибок, связанных с организацией взаимного исключения, сводится к минимуму.

10.4. Управление оперативной памятью

Память для процесса является таким же важным ресурсом, как и процессор. Процесс может выполняться только в том случае, если все необходимые данные находятся в оперативной памяти. Управление оперативной памятью предполагает решение следующих задач:

распределение физической памяти ОЗУ между процессами;

программная поддержка виртуальной памяти;

подкачка;

защита памяти.

Конкретные алгоритмы зависят от свойств применяемой ЭВМ. Для абстрактной ЭВМ рассмотрим страничную организацию ОЗУ, включающего в себя N физических страниц. Система команд машины позволяет адресовать до k страниц памяти. В этом случае частичные действия абстрактной ОС по управлению ОП (рис. 10.3) следующие.

Операционная система формирует таблицу страниц (ТС).

Каждая строка ТС содержит информацию о статусе соответствующей физической страницы, т. е. свободна она или принадле-

1.1	ТС
0	0
1 процесс *j*	1 процесс *j* контекст *j*
2 процесс *j*	2 процесс *j*
3 процесс *j*	3 процесс *j*
4 процесс *m*	4 процесс *m* контекст *m*
.....	
I процесс *j*	*I* процесс *j*
1 свободен	1 свободен
.....	
N–1	*N*–1

Рис. 10.3. Таблица страниц

жит *j*-му процессу (в этом случае в строке помещается ссылка на контекст соответствующего процесса).

Для каждого процесса, обрабатываемого в системе в данный момент времени (размещенного в БОП), ОС формирует программные структуры данных, в которых размещается информация контекста. Среди прочих значений в контексте размещается таблица страниц процесса (ТСП). Из нее можно получить данные об используемых в процессе виртуальных страницах и их месторасположении. Под *месторасположением* понимают соответствие виртуальной страницы некоторой физической странице или указание координат места на ВЗУ, где размещена копия данной страницы.

Соответственно поддержка решения задач управления ОП будет следующая. При поступлении процесса в БОП заполняется ТСП. В начальный момент из описателей процесса, сформированных на этапе обработки в БВП, выбирается список виртуальных страниц, который размещается в ТСП. Затем ОС анализирует содержимое ТСП и «приписывает» виртуальным страницам их физические эквиваленты (при этом идет загрузка содержимого соответствующих виртуальных страниц из внешней памяти в физические страницы ОЗУ).

Для виртуальных страниц процесса, которым не были выделены физические страницы, в ТСП устанавливается признак отсутствия физической страницы (этот признак также будет проставлен во все строки таблицы, соответствующие виртуальным страницам, не используемым процессом). Формируется содержимое таблицы «откаченных» страниц процесса (ТОСП) (указывается номер виртуальной страницы и ее месторасположение во внешней памяти). Далее ОС из контекста данного процесса заполняет содержимое таблицы виртуальных страниц (ТВС) процессора и передает управление на начало выполнения процесса.

В рассмотренном примере затронуты элементы решения задач распределения физической памяти, поддержки использования

аппарата виртуальной памяти, подкачки страниц. На самом деле логика действий существенно сложнее.

10.5. Планирование

Важной проблемой, на решение которой ориентированы многие компоненты современных ОС, является определение задач планирования предоставления услуг или функций ОС. Традиционно эти задачи включают в себя планирование:

очереди процессов на начало обработки процессором;

распределения времени ЦП между обрабатываемыми в мультипрограммном режиме процессами;

порядка обработки заказов на обмен с ВУ;

порядка обработки прерываний;

использования ОЗУ (организация свопинга).

В целом, комплексное решение задач планирования в ОС определяет основные эксплуатационные качества каждой конкретной системы. При планировании очереди процессов на начало обработки ЦП могут применятся как примитивные стратегии организации очереди FIFO, так и стратегии, учитывающие порядок поступления в очередь и объем ресурсов, декларированных процессами для использования. В общем случае очередь процессов в БВП может предоставляться как объединение подочередей, где каждая подочередь включает в себя определенные классы процессов (например, такая классификация может строиться на объеме запрашиваемых ресурсов и/или типе процесса). При этом возможно определение приоритета каждой из очередей (сначала рассматриваются непустые очереди с наименьшим приоритетом).

При планировании распределения времени работы ЦП между отдельными процессами возникает несколько проблем:

выбор величины кванта времени работы ЦП, выделяемого выполняемому процессу;

стратегия выбора процесса, который будет выполняться ЦП, из множества процессов, готовых к выполнению и размещенных в БОП.

В случае планирования очереди запросов на обмен решаются задачи определения порядка обработки запросов и проблемы взаимосвязанных запросов на обмен.

Комплексное решение проблем планирования однозначно определяет основные эксплуатационные характеристики и тип ОС. Рассмотрим примеры реализации задачи планирования в ОС различного типа.

Пакетная ОС. Пусть имеется *пакет программ* — некоторая совокупность программ, обладающих общим свойством — для выполнения каждой из которых требуется значительное время работы

ЦП. Необходимо обработать все программы пакета за минимальное время. В этом случае используют специализированные пакетные ОС. Для данных ОС не является важным порядок выполнения программ пакета и время, за которое они были выполнены. Критерием эффективности пакетной ОС является уменьшение времени, затраченного на выполнение всего пакета за счет минимизации, в свою очередь, непроизводительной работы ЦП. Основной задачей системы планирования пакетной ОС является максимальная загрузка процессора мультипрограммным выполнением программ-процессов пользователей. В частности, должно быть минимизировано время работы ОС. Это достигается за счет стратегии планирования, основанной на переключении выполнения одной программы-процесса на другую только в одном из следующих случаев:

завершение выполнения программы-процесса;

возникновение при выполнении программы-процесса прерывания (например, обращение к ВУ);

фиксация ОС факта зацикливания процесса.

При подобной организации планирования соотношение времени работы процессора, затраченного на выполнение программ пользователей, ко времени, затраченному на выполнение функций ОС, будет максимально.

Операционные системы разделения времени. В такой системе одновременно работает некоторое число пользователей, например терминальный класс, в котором проходит практические занятия группа студентов, и каждый пользователь работает со своей копией программы редактора или транслятора (или любой другой программой). Качеством работы пользователей в этой ситуации является создание у него иллюзии, что он работает в системе один, т.е. время, в течение которого пользователь ожидает ответ системы на запрос, должно быть минимально (запросом может быть нажатие кнопки исполнения на клавиатуре или отправка на выполнение отладочной программы и т.п.). Для работы системы в таких условиях используются ОС разделения времени. Суть функционирования подобных систем заключается в следующем. В системе определено понятие *кванта времени ЦП* — некоторый фиксированный промежуток времени работы ЦП. Каждому выполняющемуся в системе процессу выделяется квант времени ЦП, переключение выполнения на другой процесс осуществляется:

при исчерпании процессом выделенного кванта времени;

завершении выполнения программы-процесса;

возникновении при выполнении программы-процесса прерывания (например, обращение к ВУ);

фиксации ОС факта зацикливания процесса.

Характеристики конкретной системы разделения времени зависят от деталей стратегии распределения квантов ЦП, их величины, критериев выбора очередного процесса для выполнения.

Операционные системы реального времени. Существует класс задач компьютерного управления теми или иными техническими объектами. Спецификой этих задач является реакция на события, возникающие при управлении, в сроки, когда эта реакция имеет смысл (если кто-то звонит по телефону, то имеет смысл поднять трубку до того времени, пока звонящий не опустил свою трубку, поднятие трубки позднее — бессмысленно). Примеров подобных задач множество — от сложных и крайне ответственных, например управления автопилотом самолета, до прозаических и менее ответственных задач, например управления посудомоечной машиной. В общем случае все подобные задачи имеют фиксированный набор некоторых событий, реакция на произвольное возникновение и обработка которых должна быть осуществлена за некоторое гарантированное время (для каждого события это время может быть своим).

Операционная система является системой реального времени, если она при функционировании может обработать возникновение любого из данных событий (прерываний) за время, не превосходящее некоторое предельное значение. Системы реального времени являются специализированными системами, в которых все функции планирования ориентированы на достижение поставленной цели.

10.6. Файловые системы

Файловая система (*ФС*) — часть ОС, представляющая собой совокупность организованных наборов данных, хранящихся на внешних запоминающих устройствах, и программных средств, гарантирующих именованный доступ к этим данным и их защиту. Данные называются *файлами*, их имена — *именами файлов.*

Файловые системы классифицируют по степени персонификации доступа к содержимому файлов на однопользовательские и многопользовательские файловые системы.

В *однопользовательской* ФС не регламентируется доступ к содержимому файлов от имени любого пользователя. *Многопользовательские* ФС предусматривают работу только идентифицированных системой пользователей; их основным свойством является наличие защиты данных, содержащихся в файлах, от несанкционированного доступа.

Файлы обладают следующими свойствами:

1. Файл представляет собой некую сущность, имеющую имя и позволяющую оперировать со своим содержимым через ссылку на имя файла.

2. Реальное месторасположение данных файлов определяется файловой системой и в общем случае срыто от пользователя.

3. Определен фиксированный программный интерфейс для работы с содержимым файла. Операционная система однозначно определяет набор функций, обеспечивающих обмен с файлом. Этот набор функций содержит следующие возможности по работе с файлами:

1) *открытие файла*. Эта функция обеспечивает установление взаимосвязи между программой и хранящимся на внешнем носителе файлом. Это средство объявляет ОС тот факт, что с данным файлом будет работать тот или иной процесс. Операционная система, исходя из этой информации, может принять решения по изменению статуса файла (например, блокировать, разрешить или синхронизировать доступ к этому файлу со стороны других процессов и т. п.). При открытии файла система формирует внутренние наборы данных, необходимые для работы с содержимым файла. С этими наборами данных ассоциируется понятие *файлового дескриптора*;

2) *закрытие файла*. Закрытие файла — информация ОС о том, что работа с файлом завершена. При этом меняется статус доступа к файлу со стороны процессов. Операция закрытия файла осуществляется двумя функциями — закрыть и сохранить текущее содержимое файла; уничтожить файл;

3) *создание нового файла*. Функция создает новый файл. В некоторых ОС создание файла осуществляется по функции открытия файла;

4) *чтение-запись*. Обычно обмен с файлами производится некоторыми блоками данных. С одной стороны, размеры этих блоков данных могут варьироваться программистом, с другой стороны, реальные физические ресурсы имеют блочную структуру и, следовательно, определенный размер блока. Поэтому эффективность обменов, а следовательно, и эффективность работы всей ВС в целом в данном случае зависит от квалификации программиста. В современных ОС заложены механизмы сглаживания подобного рода несоответствий;

5) *управление файловым указателем*. С каждым открытым файлом связано понятие *файлового указателя*. Этот указатель в каждый момент времени показывает на следующий относительный адрес по файлу, с которым можно произвести обмен. После обмена с данным блоком указатель переносится на позицию через блок. Для организации работы с файлами необходимо уметь управлять этим указателем. В ОС определена функция, позволяющая произвольным образом перемещать указатель в пределах файла. Доступ к содержимому файла может быть прямым или последовательным. Файловый указатель — это некоторая переменная, доступная программе, которая создается при открытии файла.

4. Персонификация и защита данных. Персонификация — возможность системы «опознавать» конкретного пользователя и ас-

социировать с ним его файлы. Защита доступа к содержимому файлов обычно включает в себя назначение и проверку прав на выполнение чтения, записи и исполнения содержимого как процесса. Отметим, что персонификация и защита данных — свойство всей ОС в целом.

Одноуровневая организация ФС с непрерывными сегментами. На внешнем запоминающем носителе выделяется некоторая непрерывная область. Данные размещаются в подряд идущих единицах этого носителя. В этой области, в свою очередь, выделяется подобласть для хранения информации о файлах, которая называется *каталогом.*

Каталог представляет собой таблицу, имеющую три колонки: имя файла, координаты начала и конца файла, указанные в блоках (табл. 10.1). Имя файла в таблице должно быть уникальным (отсюда и термин — «одноуровневая»). При создании файла в эту таблицу добавляется строка с перечисленными выше характеристиками. В случае уничтожения соответствующая строка удаляется из таблицы. Функция открытия уже существующего файла сводится к нахождению в каталоге имени файла, определении его начала и конца. Операции чтения-записи происходят почти без дополнительных обменов, так как при открытии файла пользователь получает диапазон размещения данных (более того каталог можно хранить в оперативной памяти). К несомненным достоинствам одноуровневой организации относятся простота реализации и эффективность операций обмена.

Особенностью этой организации является физическая непрерывность файла на внешнем носителе. Следовательно, при создании файла необходимо знать его максимальный размер, так как в случае необходимости добавления данных в файл может не оказаться достаточно места. Система может отказать в выполнении такой операции или попытаться найти новую непрерывную область достаточного размера, чтобы переписать туда данные. Последняя операция трудоемка и приведет к внешней фрагментации, когда появятся незаполненные отрезки памяти между файлами. Выделение файлу места «заведомо достаточного» приведет к внутренней фрагментации. Можно осуществлять регулярную компрессию данных, но это операция также достаточно трудоемка.

Таблица 10.1

Пример каталога

Имя	Начало	Конец
cc	2352	2376
bp	2934	8024
delphi	15 243	15 678
...	...	...

Такого рода организация может быть пригодна для однопользовательской файловой системы. Это объясняется следующим. При большом числе пользователей очень быстро произойдет фрагментация, а постоянный

запуск компрессии приведет к неэффективности работы системы; кроме того, ограниченность числа файлов в каталоге и необходимость уникального именования файлов делают многопользовательский режим крайне неудобным.

Файловая система с блочной организацией файлов. В данном случае на пространстве внешней памяти выделяется непрерывная область данных, в которой размещается каталог. Вся оставшаяся внешняя память, предназначенная для файловой системы, разбивается на блоки, удобные для обмена. Число строк в каталоге совпадет с числом этих блоков. Каждая строка таблицы соответствует *i*-му блоку файловой системы. Каждый файл занимает, как минимум, один блок памяти. Таблица разбивается на столбцы. Поле «имя» содержит имя файла, который занимает данный блок памяти, а поле «атрибуты» — различные подполя: имя пользователя, номера блоков, занимаемых файлом. Блоки, принадлежащие одному файлу, физически могут располагаться в произвольном порядке. Такой способ организации файловой системы решает проблему лимитирования размера файла. При создании нового файла его размер будет всегда равным одному блоку, если при записи в файл его фактический размер превышает размер одного блока, то в таблицу записывается новая строка с именем этого файла и соответствующими атрибутами. Это означает, что к файлу присоединяется еще один блок и т.д. Незаполненные строки таблицы образуют список свободной памяти. При такой организации исчезает проблема внешней фрагментации, но актуальна проблема внутренней фрагментации. Так как даже если реальный размер файла равен одному байту, он все равно займет целый блок памяти. Уменьшение размера блока уменьшает внутреннюю фрагментацию, но увеличивает размер таблицы. Последнее усложняет работу с ней. Например, при открытии файла необходимо просмотреть всю таблицу, чтобы определить все блоки файла. Если таблица не хранится в оперативной памяти, то увеличивается число необходимых обменов.

Иерархические файловые системы. За основу логической организации такой файловой системы берется дерево (рис. 10.4). В корне дерева находится корень файловой системы — каталог нулевого уровня, в котором могут находиться либо файлы пользователей, либо каталоги первого уровня. Каталоги первого и следующих уровней организуются по аналогичному принципу. Файлы пользователя в этом дереве представляются листьями. Пустой каталог также может быть листом. Таким образом, формируется древовидная структура файловой системы, где в узлах находятся каталоги *B*, *C*, *D*, а листьями являются либо файлы, либо пустые каталоги.

Остановимся на правилах именования в иерархической файловой системе. В данном случае используется механизм, основанный

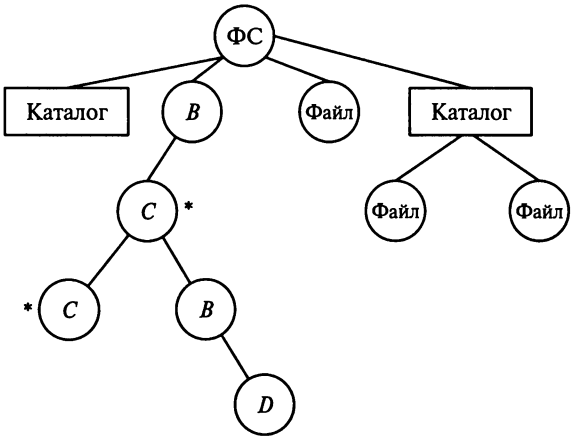


Рис. 10.4. Организация файловой системы

на понятии имени файла (name) и полного имени файла (path_name). *Полное имя файла* — это путь от корневого каталога до листа (такой путь всегда будет уникальным). Существует также относительное именование, т.е. когда нет необходимости указания полного пути при работе с файлами. Это происходит в случае, когда программа вызывает файл и подразумевается, что он находится в том же каталоге, что и программа. В данном случае появляется понятие текущего каталога, т.е. каталога, на работу с которым настроена файловая система в данный момент времени. В рамках одного каталога имена файлов одного уровня должны быть разными.

Согласно этой иерархии, с каждым из файлов можно ассоциировать атрибуты, связанные с правами доступа. Правами доступа могут обладать как файлы, так и каталоги. Структура этой системы удобна для организации многопользовательской работы за счет приведенного выше способа именования и возможности удобного наращивания ФС.

В настоящее время на практике используются различные ОС, в которых по-своему реализованы рассмотренные выше функции.

Контрольные вопросы

1. Какова роль ОС в организации вычислительного процесса?
2. Какие основные функции выполняет ОС?
3. Что понимается под виртуальной машиной?
4. В чем состоят задачи ОС при управлении ресурсами вычислительной системы?
5. Что пронимается под словом «процесс»?
6. Что такое разделяемый ресурс?

7. Каковы основные логические функции ОС?

8. Какие основные задачи выполняет подсистема управления процессами?

9. Перечислите основные типы процессов.

10. Что понимается под параллельными процессами? Как организуется их взаимодействие?

11. Поясните основные задачи управления памятью.

12. В чем состоит сущность задач планирования ОС?

13. Какие основные типы ОС вы знаете? Дайте их характеристику.

14. Что представляет собой файловая система?

15. Какие виды файловых систем существуют?

16. Какими свойствами обладают файлы?

17. Какие существуют функции по работе с файлами?

18. Дайте характеристику типовых подходов к организации файловых систем.

ГЛАВА 11

ЛОКАЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ

11.1. Общие сведения

Рост производства, усложнение социально-экономических отношений требовали совершенствования методов управления производством.

В сочетании с территориальной децентрализацией, увеличением числа людей, вовлекаемых в процесс принятия решений, отсутствие эффективных методов коммуникации, распределенного доступа к информации, автоматического ее сбора, обработки и хранения тормозили развитие экономики. Те же проблемы возникали в области управления государством.

Старые технологии сбора и обработки информации предусматривали получение данных на местах, заполнение специально разработанных форм, которые собирались в соответствующих управленческих структурах, где они проходили предварительную обработку, после чего отправлялись далее по инстанции. В результате получение окончательного результата существенно затягивалось; нередко данные обработки терялись при передаче, что приводило к значительным потерям.

Одновременно с фантастической скоростью росли возможности компьютеров, уменьшались их размеры, совершенствовались технические характеристики.

Слияние компьютеров со средствами передачи данных коренным образом изменило представление об организации ВС. Появились сети ЭВМ.

Под термином «сеть ЭВМ» обычно понимают множество соединенных между собой автономных машин. Сеть ЭВМ коренным образом отличается от распределенной системы, работая с которой пользователь не имеет ни малейшего представления, на каких процессорах и с использованием каких конкретных физических ресурсов будет исполняться его программа. В сети все машины автономны, поэтому пользователь должен делать все явно. И там, и там происходит передача информации. В сети ее инициирует пользователь, в распределенной системе — сама система. Основное различие между сетью ЭВМ и распределенной системой лежит в организации их ПО.

11.2. Организация вычислительных сетей

Все оборудование вычислительной сети можно разделить на терминалы (или абонентские машины) и систему передачи данных. Система передачи данных состоит из каналов, каналообразующей аппаратуры, коммуникационных машин и средств сопряжения систем передачи данных (рис. 11.1).

Вычислительная сеть состоит из множества машин, на которых пользователи запускают свои приложения. Эти машины часто называют *абонентскими* или *хост-машинами* (от *англ.* host — множество). Терминалы и абонентские машины обеспечивают интерфейс пользователей и работу приложений в сети. Система передачи данных служит для взаимодействия приложений и пользователей в сети. Назначение коммуникационной подсети — обеспечить передачу данных от одного хоста к другому.

Коммуникационная подсеть состоит из коммутирующих элементов и коммуникационных каналов (их также называют линиями, магистралями). *Коммутирующие элементы* — это специализированные компьютеры, соединяющие два и более коммуникационных канала. *Каналы передачи данных* — это линии связи самой различной природы: обычные телефонные линии, представляющие собой витую пару медных проводов, коаксиальный или волоконно-оптический кабель, а также беспроводные радиочастотные каналы связи, получившие распространение с появлением мобильных абонентских станций, построенных на базе ноутбуков и персональных цифровых секретарей. Они используются в автомобилях, самолетах, гостиницах и т.д.

Важным понятием в области вычислительных сетей является «топология сети». Топология определяет характер соединения линий связи и коммутирующих элементов. Наиболее распространенными топологиями являются звезда, кольцо, дерево, полносвязанная, разряженная (рис. 11.2).

По мере распространения вычислительных сетей появляется необходимость в межсетевом взаимодействии — обмене информацией между абонентскими станциями, находящимися в раз-

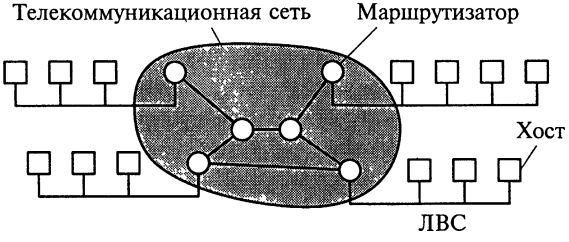


Рис. 11.1. Структура локальной сети

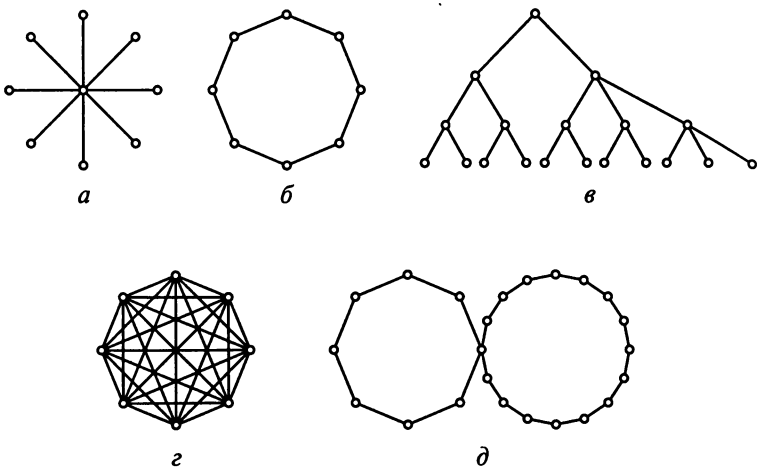


Рис. 11.2. Топологии сетей:
а — звезда; *б* — кольцо; *в* — дерево; *г* — полносвязанная; *д* — разреженная

личных сетях и использующими различные системы передачи данных (СПД). Для решения этих задач используются специальные сетевые устройства — мосты и шлюзы, т.е. средства сопряжения СПД на разных уровнях. Мост служит для соединения двух однородных СПД, шлюз — двух разных по архитектуре. Шлюз представляет собой компьютер с соответствующим ПО, обеспечивающим связь между сетями и необходимое преобразование форматов передаваемых данных. Множество соединенных сетей называют объединенной сетью. Примером такой объединенной сети может служить набор локальных вычислительных сетей, соединенных через глобальную вычислительную сеть. Глобальная сеть Интернет представляет собой такую объединенную сеть.

11.3. Классификация сетей ЭВМ

Общепризнанной таксономии сетей в настоящее время не существует. Есть два фактора для их различения: технология передачи и масштаб.

Основными типами технологий передачи, используемыми в сетях, являются: от одного ко многим (вещание) и «точка-точка».

Сети типа *вещание* имеют единый канал передачи данных, который используют все машины сети. Короткое сообщение, отправленное какой-либо машиной, называемое кадром и имеющее специальную структуру, получают все машины сети. В определенном поле кадра указан адрес получателя. Каждая машина проверяет это поле. Если она обнаруживает в этом поле свой адрес, то при-

ступает к обработке кадра. В противном случае она его игнорирует. Сети этого типа, как правило, имеют режим, когда один кадр адресуется всем машинам в сети. Это так называемый режим широкого вещания. Используется также режим группового вещания, когда один и тот же кадр получают машины, принадлежащие к определенной группе в сети.

Связи «точка-точка» соединяют пару машин индивидуальным каналом. В сети с такими связями пакет данных прежде, чем достигнет адресата, проходит через несколько промежуточных машин и связей. В таких сетях возникает потребность в маршрутизации пакетов. От ее эффективности зависит скорость доставки сообщений и распределение нагрузки в сети.

Связь типа вещание, как правило, используется на географически небольших территориях — в ЛВС, а связь «точка-точка» — для построения крупных сетей, охватывающих большие регионы (региональных сетях).

Масштаб сети — еще один критерий для классификации сетей, в соответствии с которым можно определить следующие устройства и системы в зависимости от расстояния между процессорами:

потоковая машина (плата);

многомашинный комплекс (система);

локальная сеть (комната, здание, комплекс);

региональная (глобальная) сеть (страна, континент);

Интернет (планета).

Локальная вычислительная сеть (Local Area Network — LAN) обладает следующими отличительными признаками:

размер — известна максимальная задержка при передаче;

способ передачи данных — вещание, скорость передачи 10...1000 Мбит/с;

топология — линейная, кольцо, дерево.

Региональная (*глобальная*, *широкозонная*) *вычислительная сеть* (Wide Area Network — WAN) охватывает крупные географические области, такие как страны, континенты, и обладает следующими отличительными признаками:

размер — максимальная задержка при передаче не известна;

способ передачи данных — «точка-точка»;

топология — разреженная.

Сеть этого типа объединяет множество компьютеров и локальных сетей через региональные телекоммуникационные сети.

11.4. Организация взаимодействия в вычислительной сети

Для борьбы со сложностью сеть, как правило, организована в виде иерархии слоев или уровней. В разных сетях число уровней,

их название, содержание и функции могут быть разными. Однако во всех сетях назначение каждого уровня состоит в следующем:

обеспечить определенный сервис верхним уровням;

сделать независимыми верхние уровни от деталей реализаций сервиса на нижних уровнях.

Уровень *n* на одной машине обеспечивает взаимодействие с уровнем *n* на другой машине. Правила и соглашения по установлению связи и ее поддержанию называются *протоколом* (рис. 11.3).

Фактически уровень *n* непосредственно с уровнем *n* на другой машине не взаимодействует. Он передает данные нижележащему уровню. Между каждой парой смежных уровней на одной машине существует интерфейс. Интерфейс определяет примитивы (элементарные операции) и услуги (сервис), которые нижележащий уровень должен обеспечивать для верхнего уровня.

Набор уровней и протоколов называется *архитектурой сети*. Конкретный набор протоколов, используемый на конкретной машине, называется *стеком протоколов*. Спецификация архитектуры сети должна содержать достаточно информации, чтобы разработчик сетевого ПО мог написать соответствующие программы для каждого уровня, а инженер-электронщик — создать надлежащую аппаратуру.

При передаче сообщения *M* на каждом уровне к полученным от верхнего уровня данным добавляется заголовок *H*. На уровне 2 для проверки корректности передачи данных добавляется контрольная сумма *T* — концевик. При необходимости блок данных

Рис. 11.3. Уровни, протоколы и интерфейсы

может быть разделен на несколько сегментов. Эта операция называется *фрагментацией* (на рис. 11.4 показан пример фрагментации на уровне 3). Заголовок содержит управляющую информацию — кому адресовано сообщение, время, дату, порядковый номер и т.д. Данные фактически переносятся от машины к машине по физической линии связи. На принимающей машине служебная информация (заголовок) анализируется на каждом уровне и передается на вышележащий уровень (см. рис. 11.4).

При организации уровней должны решаться следующие основные вопросы по распределению задач:

создание на каждом уровне механизма для определения отправителей и получателей;

разработка правил передачи данных между уровнями (симплекс, полудуплекс, дуплекс), а также определение числа виртуальных каналов через одно соединение и приоритетов между ними;

обнаружение и исправление ошибок;

сохранение исходной последовательности данных при передаче;

организация механизма предотвращения ситуаций, когда получатель начинает не успевать обрабатывать прибывающий поток данных;

обеспечение процессов работы на любом уровне с сообщениями произвольной длины, для чего необходимо предусмотреть разбиение, передачу и сборку сообщений; принять решение, как быть, если процесс работает со столь короткими сообщениями, что их раздельная пересылка неэффективна;

обеспечение мультиплексирования и демультиплексирования виртуальных каналов;

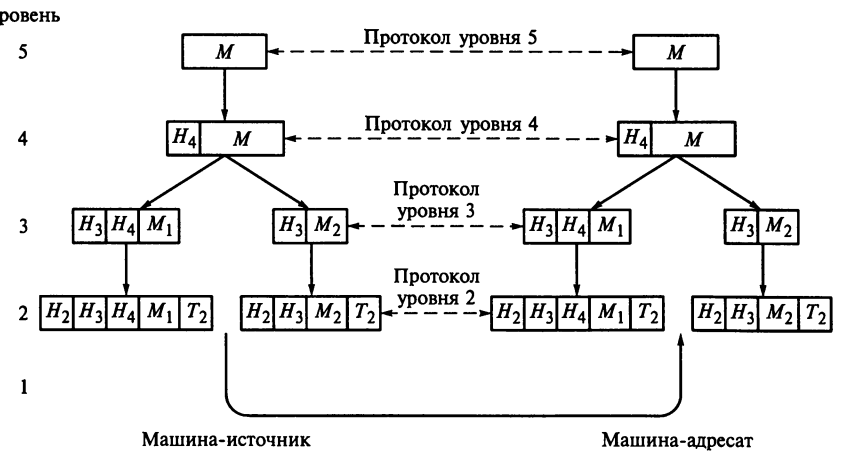


Рис. 11.4. Служебная информация, передаваемая между уровнями сети

определение правил выбора, когда между получателем и отправителем есть несколько маршрутов.

Решение всех этих вопросов производится на различных уровнях эталонной модели. Существует несколько эталонных моделей. Наиболее известна модель OSI.

Эталонная модель OSI. Модель OSI (рис. 11.5) была разработана Международной организацией по стандартизации (МОС). Эта модель имеет семь уровней

1. *Физический уровень.* Этот уровень отвечает за передачу последовательности битов по каналу связи. Основная задача уровня состоит в обеспечении достоверности передачи битов. Необходимо гарантировать, что если на одном конце линии отправили «1», то на другом получили именно «1», а не «0». На этом уровне решают такие вопросы, как способ представления «1» и «0», сколько времени тратится на передачу одного бита, следует ли поддерживать передачу данных в обоих направлениях одновременно, как устанавливается начальное соединение и как оно разрывается, каково число контактов на сетевом разъеме, для чего используется каждый контакт. Это в основном вопросы механики, электрики и оптики.

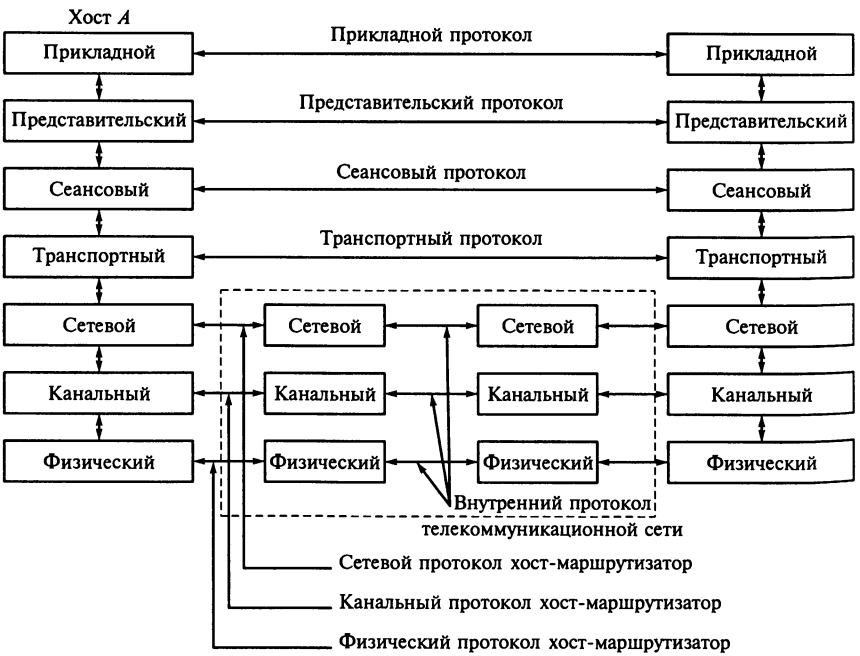


Рис. 11.5. Модель OSI

2. *Канальный уровень.* Основная задача уровня — превратить несовершенную среду передачи в надежный канал, свободный от ошибок. Эта задача решается разбиением данных отправителя на кадры (обычно от нескольких сотен до нескольких тысяч байтов), передачей кадров последовательно и обработкой кадров уведомления, поступающих от получателя. Поскольку физический уровень не распознает структуры в передаваемых данных, то определение границы кадра — задача канала данных. Она решается введением специальной последовательности битов, которая добавляется в начало и конец кадра и всегда интерпретируется как граница.

Помехи на линии могут разрушить кадр. В этом случае его необходимо передать повторно. Он будет дублирован также при потере кадра уведомления. В этом случае требуется решение новых задач на данном уровне, связанных с определением дубликатов одного и того же кадра. Уровень канала данных может поддерживать сервис разных классов для сетевого уровня, разного качества и стоимости.

Другой проблемой, возникающей на уровне канала данных (равно как и на других вышележащих уровнях), является управление потоком передачи. Например, как предотвратить «захлебывание» получателя, как сообщить отправителю размер буфера для приема передаваемых данных, имеющийся у получателя.

Если канал позволяет передавать данные в обоих направлениях одновременно, то возникает новая проблема. Кадры уведомления для потока от *A* к *B* используют тот же канал, что и трафик от *B* к *A*.

В сетях с вещательным способом передачи возникает проблема управления доступом к общему каналу. За это отвечает специальный подуровень доступа к среде (Medium Access Control — MAC).

3. *Сетевой уровень.* Этот уровень отвечает за функционирование телекоммуникационной подсети. Основной проблемой здесь является маршрутизация пакетов от отправителя к получателю. Маршруты могут быть определены заранее и прописаны в некоторой статической таблице. Они могут определяться в момент установления соединения. Наконец, маршруты могут строиться динамически в зависимости от загрузки сети. Если в подсети циркулирует слишком много пакетов, то они могут использовать одни и те же маршруты, что будет приводить к заторам. Эта проблема также решается на сетевом уровне.

Поскольку за использование телекоммуникационной сети, как правило, предполагается оплата, то на этом уровне также присутствуют функции учета — как много байт или символов послал или получил абонент сети. Если абоненты расположены в разных странах, где другие тарифы, то надо должным образом скорректировать цену услуги.

Если пакет адресован в другую сеть, то надо предпринять надлежащие меры для преобразования данных: там может быть дру-

гой формат пакетов, отличный способ адресации, размер пакетов, протоколы и т. д.

4. *Транспортный уровень.* Основная функция этого уровня — принять данные с уровня сессии, разделить, если надо, на более мелкие единицы, передать на сетевой уровень и позаботиться, чтобы все они дошли в целостности до адресата. Все это должно быть сделано эффективно и так, чтобы скрыть от вышележащего уровня непринципиальные изменения, происходящие на нижних уровнях.

В нормальных условиях транспортный уровень должен создать специальное сетевое соединение для каждого транспортного соединения по запросу уровня сессии. Если транспортное соединение требует высокой пропускной способности, то транспортный уровень может создать несколько сетевых соединений, между которыми он будет распределять передаваемые данные. И наоборот, если требуется обеспечить недорогое транспортное соединение, то транспортный уровень может использовать одно и то же сетевое соединение для нескольких соединений. В любом случае такое мультиплексирование должно быть незаметным на уровне сессии.

Сетевой уровень определяет, какой тип сервиса предоставить вышележащим уровням и пользователям сети. Наиболее часто используемым сервисом является канал «точка-точка» без ошибок, обеспечивающий доставку сообщений или байтов в той последовательности, в какой они были отправлены. Другой вид сервиса — доставка отдельных сообщений без гарантии сохранения их последовательности, рассылка одного сообщения многим в режиме вещания. Тип сервиса определяется при установлении транспортного соединения.

Транспортный уровень обеспечивает соединение «точка-точка». Модуль транспортного уровня на машине отправителя общается с равнозначным модулем на машине получателя. Нижележащие уровни общаются только с равнозначными модулями на соседних машинах, а не на машине получателя. В этом одно из основных отличий уровней 1...3 от уровней 4...7, которые только обеспечивают соединение «точка-точка».

Многие хост-машины — мультипрограммные, поэтому транспортный уровень для одной такой машины должен поддерживать несколько транспортных соединений. Для того чтобы определить, к какому соединению относиться тот или иной пакет, в его заголовке H_4 (см. рис. 11.4) помещается необходимая информация. Транспортный уровень также отвечает за установление и разрыв транспортного соединения в сети. Это предполагает наличие механизма именования, т. е. процесс на одной машине должен уметь указать, с кем в сети ему надо обменяться информацией. Транспортный уровень также должен предотвращать «захлебывание» получателя, если отправитель является очень быстрым. Такой ме-

ханизм называется *управлением потоком*. Он есть и на других уровнях. Однако управление потоком между хостами отлично от управления потоком между маршрутизаторами на уровне 3.

5. *Сеансовый уровень*. Уровень позволяет пользователям на разных машинах (пользователем может быть программа) устанавливать сессии. Сессия позволяет передавать данные, как это может делать транспортный уровень, но имеет более сложный сервис, полезный в некоторых приложениях, например вход в удаленную систему, передача файла между двумя приложениями и т. п.

Один из видов услуг на этом уровне — управление диалогом. Потоки данных могут быть разрешены в обоих направлениях одновременно либо поочередно в каждом направлении. Сервис на уровне сессии будет управлять направлением передачи.

Другим видом сервиса служит управление маркером. Для некоторых протоколов недопустимо выполнение одной и той же операции на обоих концах соединения одновременно. Для этого сеансовый уровень выделяет активной стороне маркер. Операцию может выполнять тот, кто владеет маркером. Еще одной услугой этого уровня является синхронизация. Пусть надо передать между машинами файл, пересылка которого займет два часа, а время наработки на отказ у машин составляет один час. Уровень сессии позволяет расставлять контрольные точки. В случае отказа одной из машин передача возобновится с последней контрольной точки.

6. *Представительский уровень*. Этот уровень предоставляет решения для часто возникающих проблем семантики и синтаксиса передаваемой информации и имеет дело с информацией, а не с потоком битов. Типичным примером услуги на этом уровне служит унифицированная кодировка данных. Дело в том, что на применяемых машинах используются разные способы кодировки символов (например, ASCII, Unicode), способы представления целых (в прямом, обратном или дополнительном коде), нумерация бит в байте слева направо или наоборот и т. п. Пользователи, как правило, используют структуры данных, а не случайный набор байт. Чтобы машины с разной кодировкой и представлением данных могли взаимодействовать, передаваемые структуры данных должны определяться некоторым абстрактным способом, не зависящим от кодировки, используемой при передаче. Уровень представления работает со структурами данных в абстрактной форме, преобразует это представление во внутреннее для конкретной машины и из внутреннего машинного представления в стандартное представление для передачи по сети.

7. *Прикладной уровень*. Этот уровень обеспечивает часто используемые протоколы. Например, существуют сотни разных типов терминалов. Если необходимо создать сетевой экранный редактор, то придется писать для каждого типа терминала свою версию.

Есть другой путь — определить сетевой виртуальный терминал и для него написать редактор, а затем для каждого типа терминала сделать программу его отображения на сетевой виртуальный терминал. Все ПО для виртуального сетевого терминала расположено на уровне приложений.

Другой пример — передача файлов. Разные ОС применяют разные механизмы именования, представления текстовых строк и т.д. Для передачи файлов между системами надо преодолевать все различия. Для этого используют приложение FTP (протокол передачи файлов), также расположенное на уровне приложений. На этом же уровне находятся программы электронной почты, удаленной загрузки программ, удаленного просмотра информации и т.д.

Передача данных в модели OSI. На рис. 11.6 показана последовательность действий при передаче данных в модели OSI. Хотя данные перемещаются между уровнями вертикально, каждый уровень предполагает их горизонтальное передвижение. Здесь можно провести аналогию с синхронным переводом. Когда оратор говорит на иностранном языке, он считает, что обращается к публике. Но в действительности он обращается к переводчику, а тот передает информацию слушателям.

Модель TCP/IP. Рассмотрим эталонную модель, прототипом для которой послужила сеть ARPA. С самого начала эта сеть задумывалась как объединение нескольких сетей. Одной из главных целей проекта была разработка унифицированных способов соединения сетей.

С появлением спутниковых и цифровых радиоканалов связи проблема становилась только актуальнее. Так появилась модель TCP/IP. Свое название она получила от двух основных протоколов: протокола управления передачей (Transmission Control Protocol) — TCP и межсетевого протокола (Internet Protocol) — IP.

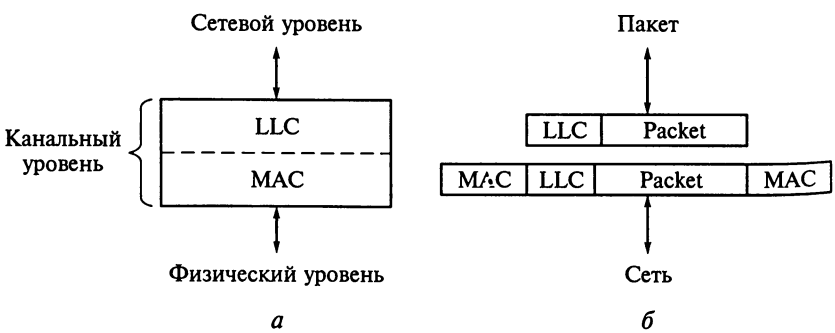


Рис. 11.6. Назначение подуровней MAC и LLC:
а — подуровни; *б* — передача пакета

Другой целью проекта ARPA было создание протоколов, не зависящих от характеристик конкретных абонентских машин, маршрутизаторов, шлюзов и т. п. Кроме этого, связь должна поддерживаться, даже если отдельные компоненты сети будут выходить из строя во время соединения, т. е. до тех пор, пока источник информации и ее получатель работоспособны и существует хотя бы один соединяющий их маршрут. Архитектура сети не должна ограничивать приложения, начиная от простой передачи файлов до передачи речи и изображения в реальном времени.

Межсетевой уровень. В силу перечисленных выше требований выбор очевиден — сеть с коммутацией пакетов с межсетевым уровнем без соединений. Этот уровень в модели TCP/IP называется межсетевым и является основой всей архитектуры. Его назначение — обеспечить доставку пакетов, движущихся в сети независимо друг от друга, даже если получатель принадлежит другой сети. Причем пакеты могут поступать к получателю не в том порядке, как они были отправлены. Упорядочить их в надлежащем порядке — задача вышележащего уровня.

Межсетевой уровень определяет межсетевой протокол IP и формат пакета. Ни протокол, ни формат пакета не являются официальными международными стандартами.

Итак, назначение межсетевого уровня в TCP/IP — доставить IP-пакет по назначению. Это как раз то, за что отвечает сетевой уровень в модели OSI.

Транспортный уровень. Над межсетевым уровнем расположен транспортный уровень. Как и в модели OSI, его задачей является обеспечение связи «точка-точка» между двумя равнозначными модулями на оконечных машинах. В рамках TCP/IP-модели было разработано два транспортных протокола. Первый протокол — TCP — надежный протокол с соединением. Он получает поток байт, фрагментирует его на отдельные сегменты и передает их на межсетевой уровень. На машине получателя модуль TCP-протокола собирает эти сообщения в поток байтов. Протокол TCP также обеспечивает управление потоком. Второй протокол — UDP (User Datagram Protocol). Это ненадежный протокол без соединения для тех приложений, которые используют свои механизмы фрагментации и управления потоком. Он часто применяется для передачи коротких сообщений между клиентским и серверным приложениями, а также там, где оперативность передачи важнее ее корректности.

Прикладной уровень. В TCP/IP-модели нет уровней сессии и представления. Разработчик сложного приложения берет на себя решение проблем этих уровней. Над транспортным протоколом располагается уровень приложений. Этот уровень первоначально включал в себя виртуальный терминал TELNET, передачу файлов FTP, электронную почту SMTP. Позднее к ним добавились: служба имен

домена DNS (Domain Name Service), отображающая логические имена хост-машин на их сетевые адреса, протокол для передачи новостей NNTP и протокол для работы с гипертекстовыми документами в сети Интернет HTTP.

Под межсетевым уровнем в модели TCP/IP располагается интерфейс хост-сеть. Модель ничего не говорит о том, как осуществляется связь, а лишь определяет, что хост-машина должна быть связана с телекоммуникационной подсетью через некоторый протокол (локальных или глобальных сетей). Никаких ограничений на этот протокол, равно как и рекомендаций, эта модель не накладывает.

Сравнение моделей MOC и TCP/IP. Обе модели имеют много общего. Они имеют уровневую иерархию, поддерживают понятие стека протоколов. Назначение их уровней примерно одинаково. Все уровни от транспортного и ниже используют протоколы для поддержки взаимодействия типа «точка-точка», не зависящего от организации сети. Все уровни выше транспортного ориентированы на приложения.

Модель OSI доказала свою эффективность как методологический инструмент, стала популярной, чего нельзя сказать о ее протоколах. С TCP/IP все наоборот — модели по существу нет, зато протоколы получили очень широкое распространение.

11.5. Особенности структурной реализации ЛВС

Локальные вычислительные сети располагаются на сравнительно небольшой территории (в комнате, здании, нескольких близко расположенных зданиях). Все компьютеры ЛВС подключены к общему коммуникационному каналу, через который передача данных происходит с использованием вещательной технологии. Такая организация ЛВС определяет ряд особенностей в их построении.

Архитектура ЛВС определяет организацию канального и физического уровней в рамках модели OSI. Существует несколько рабочих групп комитета IEEE 802 Международного института инженеров по электронике и электротехнике, выполняющего стандартизацию в области ЛВС:

802.1 — введение и архитектура ЛВС;

802.2 — управление логической связью;

802.3 — ЛВС с множественным доступом, контролем несущей и обнаружением коллизий (Ethernet);

802.4 — шина с маркером;

802.5 — кольцо с маркером;

802.6 — сеть с двумя шинами и двумя потоками;

802.7 — Совет по широкополосным технологиям;

802.8 — Консультационный совет по оптическим технологиям;
802.10 — виртуальные ЛВС и безопасность;
802.11 — беспроводные ЛВС (WI-FI);
802.12 — сеть с доступом по приоритету запроса;
802.14 — кабельные модемы;
802.15 — персональные (частные) сети;
802.16 — широкополосные беспроводные сети;
802.17 — кольцо с динамическими пакетами.

Основной вопрос в сетях вещательного типа с общим каналом заключается в том, чтобы распределять единственный канал между многими конкурирующими пользователями. Для решения этой задачи канальный уровень был разделен на два подуровня: управления логической связью (Logical Link Control — LLC) и доступа к среде (Medium Access Control — MAC).

Подуровень (и соответствующие протоколы) LLC обеспечивают сервис типа «точка-точка» вышележащим (обычно сетевым) процессам на двух машинах, соединенных общим каналом, вне зависимости от используемой архитектуры ЛВС. Подуровень (и соответствующие протоколы) MAC обеспечивает совместный доступ к общей среде передачи данных (медный или оптоволоконный кабель, радиочастотный канал).

К настоящему времени стандартизовано несколько архитектур ЛВС. Часть из них уже практически не используется (802.4 — шина с маркером, 802.5 — кольцо с маркером), часть находит ограниченное применение (802.6 — сеть с двумя шинами и двумя потоками, 802.12 — сеть с доступом по приоритету запроса, 802.17 — кольцо с динамическими пакетами). Другие же архитектуры (802.3 — ЛВС с множественным доступом, контролем несущей и обнаружением коллизий, или Ethernet, 802.11 — беспроводные ЛВС, или WI-FI, 802.15 — персональные, или частные, сети, 802.16 — широкополосные беспроводные сети) интенсивно развиваются и завоевывают рынок.

Рассмотрим наиболее популярные на сегодняшний день технологии 802.3 — ЛВС с множественным доступом, контролем несущей и обнаружением коллизий (Ethernet) и 802.11 — беспроводные ЛВС (WI-FI).

11.6. Локальные вычислительные сети архитектуры Ethernet

Локальные вычислительные сети архитектуры Ethernet определены рабочей группой IEEE 802.3 и являются, по существу, «прародителем» всех технологий локальных сетей. (Сеть имеет интересную историю. Начало положила беспроводная сеть ALOHA, развернутая в 1970-х гг. на Гавайских островах. Позднее фирма Xerox

построила первую локальную сеть с множественным доступом к среде, контролем несущей и обнаружением коллизий CSMA/CD, объединившую 100 персональных компьютеров на кабеле длиной в 1 км, производительность которой составляла 2,94 Мбит/с. Эта система была названа сетевым эфиром Ethernet.)

Первая сеть Ethernet имела такой большой успех, что фирмы Xerox, DEC и Intel решили объединить свои усилия и создали сеть Ethernet с производительностью 10 Мбит/с. Эта технология и составила основу стандарта IEEE 802.3.

Протокол доступа к среде. Согласно протоколу станция, прежде чем начать передачу, определяет состояние канала, т.е. прослушивает канал. Если канал занят, то она находится в состоянии ожидания. Как только канал освободился, она пытается начать передачу. При этом станция одновременно сравнивает отправленный сигнал с сигналом, принимаемым из линии. Если при этом произошла коллизия (две или более станций передают сигнал в кабель одновременно, соответственно переданный и принятый сигналы отличаются), станция сразу прекращает передачу полезной информации и оправляет так называемую jam-последовательность. Это делается для того, чтобы как можно раньше сообщить конфликтующим машинам о возникшей коллизии и предотвратить бесполезную трату времени на передачу искаженной информации. После этого каждая машина ожидает в течение случайного интервала времени и все начинается сначала. Этот протокол получил название протокола множественного доступа с контролем несущей и обнаружением коллизий (CSMA/CD).

Кабельная система IEEE 802.3. Всего стандарт допускает использование четырех категории кабелей (табл. 11.1). Исторически первым был кабель типа 10Base5 — «толстый Ethernet». Это желтого цвета кабель с отметками через каждые 2,5 м, указывающими место подключения рабочей станции. Подключение выполняется через специальные розетки с трансивером (приемником-передатчиком), которые монтируются непосредственно на кабель. Запись 10Base5 означает, что кабель обеспечивает пропускную способность 10 Мбит/с, использует аналоговый сигнал и максимальная длина сегмента составляет 500 м.

Затем появился кабель типа 10Base2 — «тонкий Ethernet». Это более простой в использовании кабель с подключением через байонетный коннектор. Он дешевле, но его сегмент не должен превосходить 200 м и содержать более 30 компьютеров.

Необходимость поиска обрывов или частичного повреждения кабеля привели к созданию совершенно иной кабельной конфигурации, использующей специальный вид кабеля — витую пару (Unshielded Twisted Pair — UTP).

Здесь каждая машина соединена кабелем в виде витой пары со специальным устройством — концентратором (hub) с помощью

Таблица 11.1

Стандарты кабельной системы IEEE 802.3

Тип	Кабель	Максимальная длина сегмента, м	Число узлов в сегменте	Примечания
10Base5	Толстый коаксиальный	500	100	Практически не используются
10Base2	Тонкий коаксиальный	185	30	Не использует концентратор
10Base-T	Витая пара	100	1024	Наиболее дешевая система
10Base-F	Оптоволокно	2000	1024	Наилучшая для связи между зданиями

стандартного соединителя RJ-45. Такая кабельная система получила название 10Base-T. В 10Base5 трансивер размещается прямо на кабеле. Он выполняет контроль сигнала несущей и обнаружение коллизий. Когда трансивер обнаруживает коллизию, он передает в кабель специальный сигнал, чтобы как можно быстрее сообщить другим трансиверам о возникшей коллизии. Трансивер, установленный на кабеле, соединяется с компьютером с помощью трансиверного кабеля, длина которого не должна превосходить 50 м. Он состоит из пяти витых пар. Две служат для передачи данных между компьютерами, две — управляющей информации от и к компьютеру и пятая пара — для подачи питания на трансивер. Некоторые трансиверы позволяют подключать к себе до восьми машин.

Трансиверный кабель подключается к контроллеру, находящемуся в компьютере. Контроллер отвечает за прием кадров и их отправку, проверку и формирование контрольной суммы. В некоторых случаях он выполняет управление буферами на канальном уровне, очередью буферов на отправку, прямым доступом к памяти машины и другие задачи. В кабельной системе 10Base2 трансивер расположен в контроллере. Каждая машина должна иметь свой индивидуальный трансивер.

Для того чтобы увеличить длину сегмента, используются *повторители*, устройства физического уровня, которые отвечают за побитное восстановление, усиление и передачу сигнала.

В кабельной системе 10Base-T трансивер отсутствует. Машина соединяется с концентратором (рис. 11.7) витой парой, длина которой не должна превосходить 100 м. Концентратор представляет собой многопортовый повторитель. Сигнал от передатчика T_x сетевого адаптера рабочей станции поступает в один из портов концентратора. Здесь он принимается приемником R_x и трансли-

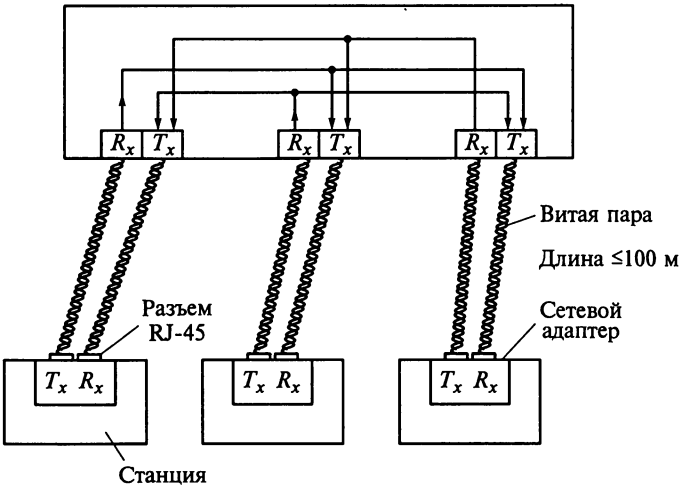


Рис. 11.7. Подключение компьютеров посредством кабеля 10Base-T

руется на передатчики всех портов. Благодаря этому для подключенных компьютеров «образуется» общая шина («один передает, все могут слышать»). Вся логика управления доступом сосредоточена в концентраторе. Повторитель обнаруживает коллизию в сегменте в случае одновременного приема сигналов нескольким приемникам R_x и посылает через все свои передатчики T_x прерывающую jam-последовательность.

Последний вид кабеля 10Base-F — оптоволоконный. Он относительно дорог, но низкий уровень помех и значительная длина одного сегмента — важные достоинства этого кабеля. Кабельные оптоволоконные линии подключаются к концентратору с оптическими портами.

В сетях Ethernet действует правило четырех повторителей — между любыми двумя компьютерами должно быть не более четырех повторителей (концентратор представляет собой один из них), а из пяти кабельных сегментов (на коаксиальном кабеле) только три могут содержать подключенные компьютеры. В случае смешанной структуры (витая пара, оптоволокно и коаксиальный кабель) для проверки возможности корректной работы сети проводят расчеты максимального времени распространения сигнала по кабельной системе.

Кодирование на физическом уровне. Прямое кодирование неоднозначно, и поэтому не используется. Нужен был метод, который бы позволял определять начало, середину и конец передачи каждого бита без специальной побитной синхронизации. Для этого было предложено два метода кодирования: манчестерский и дифференциальный манчестерский коды.

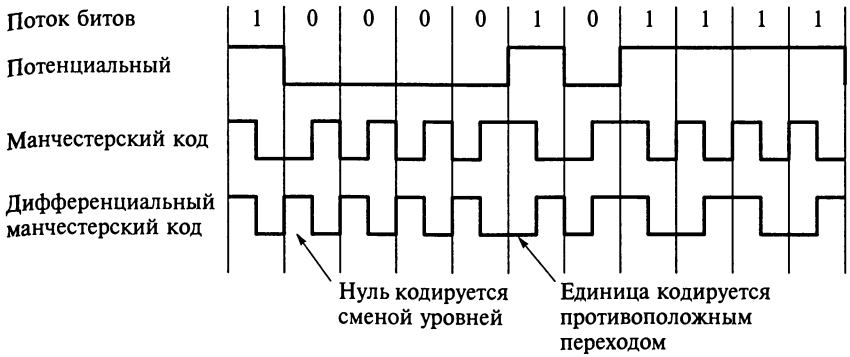


Рис. 11.8. Манчестерский и дифференциальный манчестерский методы кодирования

При использовании *манчестерского* кода интервал передачи бита разбивается на два равных интервала (рис. 11.8). При передаче «1» в первом интервале передается сигнал высокого уровня, а во втором — низкого. При передаче «0» все наоборот. При таком подходе в середине передачи каждого бита имеется переход, что позволяет синхронизовать приемник. Недостатком такого подхода является то, что пропускная способность канала уменьшается вдвое по сравнению с потенциальным кодированием, но этот недостаток вполне компенсируется возможностью самосинхронизации.

Дифференциальный манчестерский код отличается тем, что при передаче «1» в первом интервале сохраняется уровень предыдущего, а при передаче «0» — этого не делается (см. рис. 11.8).

Производительность ЛВС с множественным доступом, контролем несущей и обнаружением коллизий. Рассмотрим производительность ЛВС с множественным доступом, контролем несущей и обнаружением коллизий при условии плотной и постоянной нагрузки. Пусть всегда готовы к передаче k станций. Введем некоторые допущения для упрощения анализа. Разобьем время на равные по длительности отрезки длиной $2t$, где t — максимальное время распространения сигнала из конца в конец кабельного сегмента, и вместо анализа коллизий будем рассматривать постоянную вероятность повторной передачи кадра на каждом временно́м отрезке. Если каждая станция участвует в состязаниях с вероятностью p, то вероятность того, что некоторая станция захватит канал на этом временно́м участке,

$$A = kp(1 - p)^{k-1}.$$

Величина A достигает максимума при $p = 1/k$. Отметим, что $A \to 1/e$ при $k \to \infty$. Вероятность того, что период состязаний составит j временны́х отрезков, равна $A(1 - A)^{j-1}$.

Отсюда среднее число отрезков до завершения состязания составит

$$\sum_{j=0}^{\infty} jA(1-A)^{j-1} = \frac{1}{A}.$$

Так как каждый временной отрезок имеет длительность $2t$, то средний интервал состязаний $w = 2t/A$. Если значение p оптимально, то $w \le 2te$, что приближенно равно $5{,}4t$. Если передача кадра средней длины занимает время P, то при условии большого числа станций, постоянно имеющих кадры для передачи, эффективность канала составит $P/(P + 2t/A)$. Из этой формулы видно, что чем длиннее кабель, тем ниже эффективность, так как растет длительность периода состязаний. При длительности 51,2 мс, что соответствует 2,5 км с четырьмя повторителями и скорости передачи 10 Мбит/с, минимальный размер кадра составляет 512 бит или 64 байт.

Эффективность канала зависит от числа готовых к передаче станций при сделанных выше предположениях о длине кабеля, скорости передачи и минимальной длине кадра (рис. 11.9). С увеличением нагрузки эффективность сети падает. Эффективность сети тем выше, чем выше средняя длина кадра. Теоретически при неограниченном увеличении длины кадра эффективность стремится к 1, однако при этом также неограниченно увеличивается время ожидания доступа к каналу. Наименьшая эффективность канала наблюдается при коротких кадрах, поэтому наиболее жесткие те-

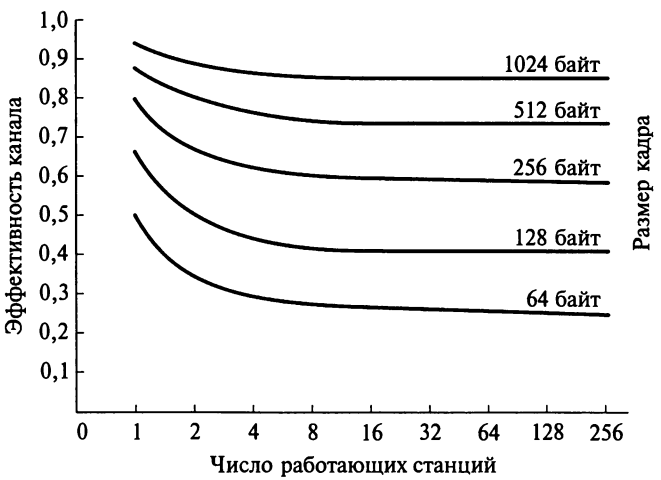


Рис. 11.9. Эффективность ЛВС с множественным доступом, контролем несущей и обнаружением коллизий

сты производительности сети проводят с использованием генераторов коротких кадров.

Технология Fast Ethernet. Эта технология представляет собой продолжение развития классической технологии Ethernet. Ее основными достоинствами являются:

увеличение пропускной способности сегментов сети до 100 Мбит/с;

сохранение метода случайного доступа Ethernet;

сохранение звездообразной топологии сетей и поддержка традиционной среды передачи данных — витой пары и оптоволоконного кабеля.

В официальном стандарте 100Base-T установлено три различных типа кабельных систем:

100Base-TX для двухпарного кабеля на неэкранированной витой паре UTP категории 5 или экранированной витой паре STP Type 1;

100Base-T4 для четырехпарного кабеля на неэкранированной витой паре UTP категории 3, 4 или 5;

100Base-FX для многомодового оптоволоконного кабеля.

Технология Fast Ethernet не требует коренного переобучения персонала и замены оборудования во всех узлах сети. Большинство сетевых адаптеров позволяют работать как в сетях Fast Ethernet, так и в Ethernet. Это возможно благодаря использованию одинаковых протоколов MAC подуровня и единого формата кадров. Отличие заключается в длительности бита: в сетях Ethernet 100 нс, а в сетях Fast Ethernet — 10 нс. Кроме того, в сетевом оборудовании Ethernet/Fast Ethernet реализована функция «автопереговоров», которая позволяет сетевым устройствам (сетевому адаптеру, концентратору, коммутатору) автоматически в процессе начального диалога определять возможности соседей и выбирать наиболее подходящий стандарт связи.

Технология Gigabit Ethernet. Следующим шагом в развитии Ethernet стало создание технологии Gigabit Ethernet, обеспечивающей увеличение скорости передачи данных до 1000 Мбит/с. В таких сетях в качестве среды передачи по-прежнему используются витая пара и оптоволокно (табл. 11.2). Допускается также использование коаксиального кабеля для связи на небольших расстояниях до 25 м, например для соединений в монтажном шкафу.

Ключевое слово в гигабитной технологии — слово «оптический». Именно оптическая проводка обеспечит высокоскоростную связь на магистралях длиной 500...5000 м. Вместе с тем для настольных станций, связанных неэкранированными медными проводами, также имеется возможность гигабитных скоростей, однако для этого потребуется использовать четыре пары проводников.

Ключевые аспекты технологии Ethernet по-прежнему остаются практически неизменными — формат кадра и метод доступа к

Таблица 11.2

Стандарты кабельной системы Gigabit Ethernet

Тип	Кабель	Максимальная длина сегмента, м	Примечания
1000Base-SX	Оптоволокно	550	Многомодовое оптоволокно (50; 62,5 нм)
1000Base-LX	Оптоволокно	5000	Одно- (10 нм) или многомодовое оптоволокно (50; 62,5 нм)
1000Base-CX	Две пары STP	25	Экранированная витая пара
1000Base-T	Четыре пары UTP	100	UTP категории 5

среде. Это и определяет популярность новой технологии, а также ее совместимость с предыдущими поколениями Ethernet.

Коммутируемый Ethernet. Традиционные средства построения ЛВС на базе концентратора или коаксиального кабеля имеют следующие ограничения по производительности и управляемости сети:

максимально допустимая длина сегмента. Она зависит от типа используемого кабеля: для витой пары — 100 м (500 м при пяти сегментах), для тонкого коаксиального кабеля — 185 м (при пяти сегментах — 925 м);

максимальное число узлов в сети. Стандарты Ethernet ограничивают число узлов в сети предельным значением в 1024 компьютера вне зависимости от типа кабеля и числа сегментов.

Существуют и другие причины, по которым число узлов в сети Ethernet обычно не превосходит нескольких десятков. Эти причины лежат в самом принципе разделения во времени одного канала передачи данных между всеми узлами сети. Даже если упрощенно считать, что все узлы получают равные доли времени при доступе к каналу и непроизводительные потери времени отсутствуют, то при наличии в сети N узлов на один узел приходится только $10/N$ пропускной способности канала. При увеличении N пропускная способность, выделяемая каждому узлу, оказывается настолько малой, что нормальная работа приложений и пользователей становится невозможной. Кроме того, поскольку запросы на доступ к среде генерируются узлами в случайные моменты времени, то при большой интенсивности запросов частота возникновения коллизий также возрастает, что приводит к неэффективному использованию канала — доля времени, в течение которого канал предоставляется в распоряжение конкретному узлу, становится еще меньше.

Для преодоления ограничений стандартных технологий локальных сетей применяют различные коммутаторы и мосты. Технология коммутации основана на отказе от применения разделяемого между всеми узлами сегмента канала связи и использовании коммутаторов, позволяющих одновременно передавать кадры между парами портов.

Коммутатор работает на канальном уровне. Он анализирует заголовки кадров, автоматически строит адресную таблицу и на основании этой таблицы направляет кадр только в один из своих выходных портов (или просто уничтожает его). Обработка кадров, поступающих от разных портов, выполняется параллельно. Если принятый кадр нужно передать на другой порт, то процессор обращается к коммутационной матрице и пытается образовать виртуальный канал, связывающий порт-источник с портом назначения. Коммутационная матрица может сделать это только в том случае, если порт назначения в этот момент свободен, т.е. не соединен с каким-либо другим портом. Если же порт занят, то кадр полностью буферизуется процессором входного порта. Процессор ожидает, пока выходной порт не освободится и коммутационной матрицей не будет образован необходимый канал. После того как канал будет образован, по нему направляется буферизованный кадр, который принимается процессором выходного порта и после получения им доступа к среде передается в подключенный к порту кабельный сегмент. Каждый порт коммутатора при передаче кадра в кабельный сегмент работает как обычный узел (сетевой адаптер) ЛВС.

Благодаря использованию в коммутаторах и мостах буферизации (в концентраторе буфер отсутствует), появляется возможность передачи кадра между сегментами, использующими различные архитектуры и скорости передачи данных (Ethernet, Fast Ethernet, FDDI, беспроводная сеть и т.д.).

Для передачи кадра из низкоскоростного сегмента в высокоскоростной необходимо принятый кадр сжать во времени. Это можно выполнить, если принятый кадр сначала записывается в буфер на одной скорости, а при передаче в другой сегмент извлекается из буфера с большей скоростью. Отсюда возникает возможность построения коммутатора с портами, работающими на разных скоростях 10,10,1000 Мбит/с. Низкоскоростные порты в таких коммутаторах используются для подключения рабочих станций, а высокоскоростные — для соединения с магистралями или высокопроизводительными серверами. Скорость работы портов определяется в ходе автопереговоров. При подключении к концентратору сетевых адаптеров, имеющих различную скорость передачи, концентратором автоматически будет выбрана наиболее низкая скорость. Таким образом, подключение к двухскоростному концентратору хотя бы одной рабочей станции с адаптером на

10 Мбит/с приведет к работе всего сегмента ЛВС на этой скорости.

К коммутатору и мосту могут подключаться сегменты с разной архитектурой. Для этого сам коммутатор или мост должен поддерживать на своих портах различные технологии ЛВС. В коммутаторе при этом выполняются дополнительные операции преобразования кадров. Например, при соединении сегментов Ethernet и FDDI (или беспроводного) кадр Ethernet принимается из одного сегмента через соответствующий порт, в коммутаторе из кадра Ethernet MAC-подуровня извлекается кадр LLC, а затем кадр LLC инкапсулируется в кадр FDDI (или беспроводной сети) MAC-подуровня и, наконец, передается в другой сегмент через соответствующий порт. При передаче сообщения между двумя рабочими станциями, находящимися в разных сегментах ЛВС, таких преобразований может быть несколько (сервер в сети FDDI, затем сегмент Fast Ethernet, беспроводной сегмент и, наконец, беспроводная рабочая станция). При этом сервер и рабочая станция, несмотря на преобразования MAC-кадров, будут взаимодействовать на канальном уровне по протоколу «точка-точка» LLC.

11.7. Беспроводные ЛВС

В последнее время все большее распространение получают беспроводные локальные сети на основе стандарта IEEE 802.11 (табл. 11.3).

Основное отличие ЛВС различных категорий заключается в способах организации физического уровня, т.е. рабочих частотах и видах модуляции. В результате обеспечивается совместимость только сетей стандартов 801.11b и 802.11g. Вместе с тем большинство

Таблица 11.3

Категории ЛВС

Характеристика	802.11	802.11b	802.11a	802.11g
Скорость передачи максимальная, Мбит/с	2	11	54	54
Средняя скорость, Мбит/с	1	4...5	27	20...25
Рабочие частоты, ГГц	2,4	2,4	5	2,4
Полоса пропускания, МГц	83,5	83,5	300	83,5
Модуляция	FHSS	DSSS/CCK	OFDM	DSSS/OFDM
Число каналов / не перекрывающихся	11/3	11/3	12/8	11/3

производителей выпускает оборудование (беспроводные точки доступа и сетевые адаптеры), в которых реализовано одновременно несколько стандартов.

Беспроводные ЛВС могут быть построены двумя различными способами: «каждый с каждым» (AD Hoc) и с точкой доступа (Infrastructure Mode). Их основное отличие состоит в различных используемых протоколах доступа к среде, так называемых MAC-протоколах. В первом случае каждая беспроводная станция может связываться непосредственно с любой другой беспроводной станцией. Во втором случае связь происходит через специальное устройство — беспроводную точку доступа, которая выполняет функцию ретранслятора. Кроме того, точка доступа выполняет функцию моста связи с проводной сетью архитектуры Ethernet.

11.8. Функции компьютера в ЛВС

Компьютер в сети может выполнять функции рабочей станции или сервера. *Сервер* — это компьютер, предоставляющий в совместное пользование свои ресурсы другим компьютерам (прикладным и системным процессам, выполняемым на других ЭВМ). Рабочие станции получают доступ к ресурсам серверов. Роль компьютера в сети определяется видом и конфигурацией его ОС.

Различают сети с выделенным сервером и одноранговые сети. В сетях с *выделенным сервером* на одном или нескольких компьютерах установлены серверные ОС, предназначенные для высокоэффективного предоставления ресурсов этого сервера рабочим станциям. В зависимости от разделяемых ресурсов различают серверы печати, файлов, приложений, баз данных, почтовые и т.д. Один и тот же компьютер, в зависимости от конфигурации серверной ОС, может одновременно выполнять роль нескольких серверов (например, файлового и печати).

В *одноранговой* сети каждый из компьютеров одновременно служит и рабочей станцией, и сервером файлов. Обычно такие сети организуются под управлением ОС фирмы Microsoft, в которых одновременно реализованы клиентская и серверная функции. В этом случае на каждом из компьютеров открывается какой-либо ресурс для совместного пользования.

Одноранговые сети просты в организации, но, как правило, объединяют не более 5... 10 машин. Основное отличие одноранговых сетей от сетей с выделенным сервером заключается в организации системы безопасности. В сетях с выделенным сервером система безопасности, разграничивающая права пользователей, хранится централизованно на одной машине и находится в совместном пользовании. В одноранговой сети каждый из компьютеров, предоставляющий свой ресурс другим компьютерам, сам отвеча-

ет за организацию собственной безопасности. В результате, управление безопасностью сети в целом в одноранговой сети становится практически невозможным. В связи с этим при наличии в сети критической информации целесообразно использовать сеть с выделенным сервером.

К компьютерам, выполняющим функции серверов в сети, обычно предъявляют повышенные требования в части производительности и надежности. Особенности построения таких машин рассматриваются в гл. 12.

Контрольные вопросы

1. Какова причина возникновения вычислительных сетей?
2. Что понимается под вычислительной сетью?
3. Назовите основные виды топологии вычислительных сетей. Какие топологии используются в ЛВС, ГВС?
4. Приведите классификацию вычислительных сетей.
5. Чем отличается связь типа «точка-точка» от связи вещательного типа?
6. Какие типы связи применяются в ЛВС, ГВС?
7. Что такое протокол связи и интерфейс?
8. Что такое архитектура вычислительной сети и стек протоколов?
9. Каково назначение уровней в эталонной модели взаимодействия открытых систем?
10. Каково назначение уровней в сетевой модели TCP/IP?
11. Поясните причину выделения в локальных сетях подуровня управления логической связью и подуровня доступа к среде.
12. Перечислите основные архитектуры ЛВС.
13. Опишите особенности построения сети Ethernet (протокол MAC-подуровня, кабельная система, способ физического кодирования).
14. Назовите основные компоненты ЛВС Ethernet, укажите их назначение.
15. Дайте характеристику архитектурам ЛВС Ethernet, Fast Ethernet, Gigabit Ethernet.
16. В чем заключаются отличия в работе концентратора и коммутатора Ethernet?
17. С помощью каких устройств соединяются сегменты ЛВС различной архитектуры?
18. Что такое сеть с выделенным сервером?
19. Что такое сервер ЛВС?
20. Что такое одноранговая сеть?
21. В чем заключается различие в организации сети с выделенным сервером и одноранговой сети?

ГЛАВА 12

ОСОБЕННОСТИ МАШИН ДЛЯ ПОСТРОЕНИЯ СЕРВЕРОВ

12.1. Требования к серверам различного назначения

Требования к локальным сетям существенно различаются в зависимости от назначения сети. В банковских сетях клиентская машина может быть весьма простой, а вся обработка и хранение необходимой информации осуществляется сервером централизованно. В других локальных сетях, например сетях издательских домов, к клиентскому компьютеру предъявляются значительно более строгие требования. Однако это, как правило, не приводит к снижению требований к серверу сети.

Компьютеры, используемые в качестве большинства серверов в сети, должны обладать производительностью, достаточной для обслуживания всех клиентских компьютеров сети, очень большими объемами ОП для хранения всей информации и исключительно высокой надежностью.

Современные процессоры персональных компьютеров Pentium IV, AMD, а также Apple обладают достаточно высоким быстродействием, но адресуемые объемы ОП недостаточны для создания серверов в крупных сетях. Поэтому для серверов используют специальные процессоры с расширенным адресным пространством, а для обеспечения надежности информации ее хранят в дисковых массивах RAID.

12.2. Системы автоматического контроля и диагностики

Одним из важнейших требований к вычислительным системам, в особенности к серверам, является обеспечение надежности, так как любой отказ ВС может привести к тяжким последствиям. Для улучшения надежности ВС существуют многочисленные методы, применение которых определяется назначением системы, условиями ее работы и вероятными видами отказов.

Виды отказов и статистические характеристики надежности. Все ошибки в работе ВС подразделяют на систематические, возникающие в результате отказов, и случайные, вызванные сбоями. *От-*

каз — это устойчивое нарушение работоспособности аппаратуры, вызываемое, как правило, выходом из строя одного или нескольких элементов. Отказы бывают внезапными и постепенными, проявляющимися вначале в виде неустойчивых дефектов. *Сбоем* называют кратковременное самоустраняющееся нарушение правильного функционирования ВС; причиной сбоев чаще всего бывают электромагнитные и электростатические помехи, возникающие как во внешней среде, так и внутри компьютера. Помимо отказов и сбоев в аппаратуре существуют ошибки в ПО, не выявленные в процессе отладки программ. С такими ошибками обычно приходится сталкиваться при некоторых редко встречающихся сочетаниях обрабатываемых данных и командной информации.

Отказы и сбои происходят в любых компьютерах. Но особенно они опасны в устройствах, входящих в состав управляющих систем и комплексов. Они приводят к потере информации, длительным простоям или катастрофическим последствиям. Именно по этой причине проблема своевременного выявления и исправления ошибок в процессе решения задачи является актуальной.

Решение этой проблемы связано с введением в обрабатываемую информацию избыточности. Избыточность может быть двух видов — временной или пространственной. *Временна́я* избыточность связана с увеличением времени решения задачи. Например, задачу можно решать дважды на одном компьютере, используя одну или две разные программы, и сравнивать получаемые результаты. Она служит основой для программно-логического контроля. Сюда принято относить тестовые проверки, повторные просчеты, расчеты с использованием других программ и т.д. Такие методы не нуждаются в значительных аппаратных затратах, но существенно увеличивают время решения, нередко требуя более высокого быстродействия. Для сокращения затрат на повторное выполнение программы в нее включают контрольные точки, позволяющие повторять не всю, а лишь участок программы, начиная с предыдущей контрольной точки. Расстановка контрольных точек особенно сложна в мультипроцессорных и многомашинных ВС и должна предотвращать эффект «домино», т.е. распространения ошибочного результата за счет обменов между процессорами.

Один из получивших наибольшее распространение методов программно-логического контроля — тестовые проверки — основан на выполнении специальных программ с известным результатом. Глубина проверки зависит от времени, отведенного на проверку, и частоты проверок. Характерным для таких проверок служит то обстоятельство, что они дают информацию об исправности ВС на момент проверки, поэтому полной уверенности в правильности решения задачи быть не может. Интервал ΔT между тестовыми проверками можно найти исходя из формулы, определяющей вероятность безотказной работы в момент $(t + \Delta T)$, так

как во время выполнения проверки аппаратура работала безотказно: $\Delta T = \Delta P/\lambda$. Тестовые проверки являются одним из самых распространенных методов, позволяющих обнаружить постоянные отказы компьютера. Изучение методов программного контроля — задача программирования.

Пространственная избыточность заключается в увеличении длины машинного слова, в которое вводятся дополнительные разряды. Например, программа может одновременно выполняться на двух компьютерах. Если полученные результаты совпадают, то задача решена верно. Для реализации пространственной избыточности необходимы дополнительные средства, а именно: средства аппаратурного контроля. К этой группе контроля относят корректирующие коды, мажоритарные схемы и т. п. Эти методы приводят к значительным аппаратным затратам, но обеспечивают возможность непрерывной работы ВС в условиях отказов и сбоев. Они являются единственными методами, позволяющими повысить надежность необслуживаемых (например, находящихся на борту автономного космического аппарата) ВС.

При пространственной избыточности возможность обнаружения ошибок основывается на добавлении к n-разрядному двоичному слову дополнительных m контрольных разрядов. Ошибки в двоичном слове — это появление «1» вместо «0» или «0» вместо «1». Если некоторая кодовая комбинация (а всего их 2^n) перешла в разряд запрещенных, то это служит свидетельством наличия ошибки. Если в результате появления ошибки произошла замена некоторой разрешенной комбинации другой, но также разрешенной (например, при отсутствии избыточности), то такая ошибка не обнаруживается. Корректирующая способность кода, т. е. способность обнаруживать и исправлять ошибки, зависит от его избыточности. Различают абсолютную и относительную избыточность кода. *Абсолютная* избыточность представляет собой число контрольных разрядов m, а *относительная* избыточность определяется как отношение абсолютной избыточности к общей длине машинного слова:

$$C = m/(n + m).$$

Любой код, обладающий ненулевой избыточностью и позволяющий исправлять или только обнаруживать ошибки, называется *корректирующим*. В технике принято подразделять корректирующие коды на посылочные (используемые при передачах информации) и арифметические (служащие для контроля устройств обработки).

Корректирующая способность кода и функции систем контроля. Задача разработчика при выборе средств аппаратурного контроля сводится к определению такого кода, который при минимальной

аппаратной избыточности обеспечивал бы необходимую корректирующую способность.

Введем некоторые понятия. Назовем *кодовым весом* число разрядов машинного слова, содержащих «1». Теперь сравним две кодовые комбинации одинаковой длины. Часть разрядов может содержать совпадающие двоичные цифры, а часть несовпадающие. Число разрядов с несовпадающими значениями назовем *кодовым расстоянием* и обозначим через d. Кодовое расстояние можно определить, выполнив поразрядное сложение по модулю 2 этих комбинаций и определив кодовый вес полученной суммы. Иногда кодовое расстояние d называют *хеммингoвым*. Обычно интерес представляет минимальное кодовое расстояние $d_{\min}$, т.е. самое малое кодовое расстояние между двумя любыми используемыми комбинациями.

В обычном двоичном коде минимальное расстояние $d_{\min} = 1$, т.е. такой код не является корректирующим. Код называют корректирующим, если минимальное кодовое расстояние $d_{\min} \geq 2$. При $d_{\min} = 2$ корректирующий код только обнаруживает одиночную ошибку, но не может ее исправить, так как запрещенная комбинация равно удалена от двух разрешенных. Если $d_{\min} = 3$, то одиночная ошибка приводит к возникновению запрещенной комбинации, отстоящей от исходной на единицу; все остальные разрешенные комбинации будут отстоять от нее минимум на две единицы.

Следовательно, задача исправления ошибки сводится к поиску разряда, содержащего ошибку, и добавление единицы к которому приведет к исходному безошибочному коду.

Контроль передач информации. Чтобы обнаружить одиночную ошибку, достаточно добавить к двоичному слову всего один контрольный разряд. Обычно используют контроль по четности (или нечетности), т.е. в контрольном разряде проставляется «1», если сумма по модулю 2 всех разрядов исходной комбинации равна единице, и «0», если эта сумма равна нулю. Таким образом, вес любой безошибочной комбинации всегда остается четным; одиночная ошибка приводит к нечетному весу этой избыточной комбинации. Дополнительный контрольный разряд вдвое увеличивает число возможных кодовых комбинаций.

Одиночная ошибка ведет к исчезновению «1» в одном разряде исходной числовой комбинации или ее появлению в разряде, который первоначально содержал «0». Это приводит к нарушению четности и может быть выявлено проверкой на четность; для этого выполняют декодирование, т.е. сложение по модулю 2 всех разрядов слова (находят «свертку»), включая контрольный разряд. Если результат свертки равен нулю, то числовая комбинация принята без ошибок, а если единице, то в принятой комбинации присутствует ошибочный разряд.

Этот способ контроля выявляет все комбинации, в которых при передаче возникли нечетнократные (одиночные, тройные и т.д.) ошибки. Кроме того, он обладает небольшой избыточностью. Все это привело к тому, что он находит широкое применение для контроля правильности межрегистровых передач и работы основной памяти компьютера.

Для исправления ошибок при передачах информации можно воспользоваться той же идеей: введением дополнительных контрольных разрядов. Если имеется два контрольных разряда, то можно определить место ошибочного разряда с точностью до полуслова, т.е. указать половину слова, которая содержит ошибку. Увеличивая число контрольных разрядов, можно точно определить «ошибочный» разряд, а значит, исправить возникшую ошибку. Если этот «ошибочный» разряд содержал «0», то для исправления ошибки в него нужно записать «1», и наоборот, если в нем оказалась «1», то на ее место нужно записать «0».

Именно такой способ выявления ошибок предложил Р.Хемминг. Он последовательно разбивал слово на два полуслова (рис. 12.1). В результате такого разбиения получилось $\log_2 n$ полуслов, причем в каждом полуслове встречается один разряд, принадлежащий исключительно этому полуслову. Этот разряд и принимается в качестве контрольного. Таким образом, контрольными будут 1, 2, 4 и 8-й разряды. При другом выборе полуслов получается другой корректирующий код с иными номерами контрольных разрядов. Однако это не имеет принципиального значения, так как дешифровка кода возлагается на аппаратуру. Не охваченным проверкой остался только один нулевой разряд. Его значение может быть как ошибочным, так и верным, так как он не попал ни в одно из проверяемых полуслов. Обычно этот разряд не включают в состав проверяемого слова. Таким образом, при наличии 16 разрядов только 11 являются информационными, четыре служат для размещения контрольной информации, а последний игнорируется.

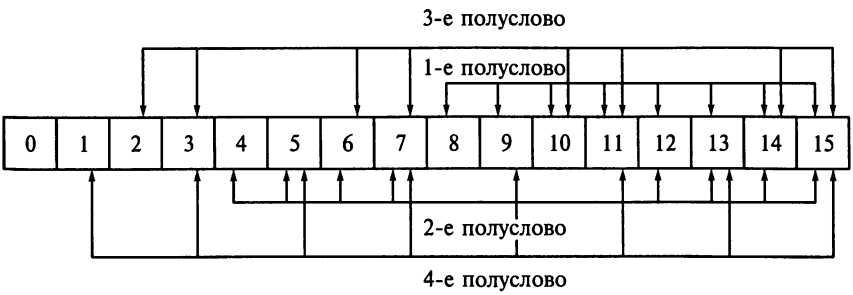


Рис. 12.1. Разбиение кодовой комбинации на полуслова

Если контрольные разряды записывать в специальный регистр, то появившееся в нем число определяет номер ошибочного разряда, и остается только произвести сложение по модулю 2 содержимого этого разряда с единицей, или инвертирование этого разряда. Код Хемминга и все ему подобные коды имеют $d_{\min} = 3$.

Контроль арифметических операций. Для проверки правильности выполнения операции сложения, а следовательно, и операций вычитания, умножения, деления и других, которые в компьютере выполняются как многократное сложение, можно воспользоваться контролем по четности, если привлечь некоторые дополнительные сведения об операндах.

Пусть нужно произвести сложение двух чисел $A + B = S = s_n, s_{n-1}, \ldots, s_1$. Значение четности ее кодового веса

$$P_s = s_n \oplus s_{n-1} \oplus \ldots \oplus s_1.$$

В результате сложения получим

$$P_s^* = P_A \oplus P_B \oplus P_C,$$

где P_A и P_B — контрольные цифры операндов A и B соответственно; P_C — контрольная цифра числа, образованного из значений переносов. Значения P_s и P_s^* при отсутствии ошибки должны совпадать, а при возникновении ошибки во время операции сложения $E = P_s^* \oplus P_s = 1$.

Проверка умножения, деления и других операций может выполняться на каждом шаге, когда производится операция сложения. Можно воспользоваться таким же приемом и для исправления ошибок при выполнении арифметических операций, если применить код Хемминга. Однако аппаратура для этого довольно громоздкая и используется обычно только в специальных компьютерах, например бортовых.

12.3. Защита памяти. RAID-массивы

В компьютерной сети очень выгодно держать всю системную информацию не в виде отдельных копий, а централизованно на диске сервера, что позволяет существенно снизить затраты. При необходимости воспользоваться той или иной частью информации компьютер обращается к серверу и перегружает нужные данные в свою память. Однако при таком подходе резко возрастает опасность полной потери информации при возможном выходе из строя дискового накопителя сервера. В настоящее время жесткие диски становятся все бóльшими по объему и более быстродействующими, но и они не застрахованы от отказов. Чтобы избежать потерь информации при выходе диска из строя, его содержимое периодически копируется на второй диск или магнитную ленту.

Это позволяет исключить полную потерю данных, накопленных за долгое время работы, но не решает проблему полностью: приходится отключать сервер для замены вышедшего из строя диска, находить магнитную ленту с соответствующей копией потерянной информации, восстанавливать содержимое диска. При этом восстановление будет неполным — восстанавливается лишь информация, сохраненная на ленте в момент ее копирования с диска, а более поздняя информация теряется. На время отключения сервера пользователи сети не имеют доступа к нужной им информации.

Для защиты данных от возможных потерь из-за отказов диска сервера предложена архитектура RAID, призванная защитить данные в реальном времени. Ее идея состоит в том, чтобы вместо одного дорогого диска большого объема, где хранится вся информация сети, воспользоваться «избыточным массивом недорогих дисков»; именно так расшифровывается и переводится этот термин. Нужно помнить, что любой RAID-массив всегда дороже одного диска большого объема и, как правило, имеет больший объем памяти. Он позволяет обеспечить более высокую производительность по сравнению с накопителем на большом диске и добиться высокой отказоустойчивости.

Массив дисков, будучи подключенным к компьютеру, воспринимается им как единый диск большого объема. Все распределение информации по отдельным дискам такого массива осуществляется единым контроллером, а отказоустойчивость обеспечивается за счет избыточности хранимой информации и нужного распределения ее по отдельным дискам. Кроме того, питание RAID-массива осуществляется от нескольких общих автономных источников питания, что также способствует повышению отказоустойчивости. Производительность RAID-массива повышается по сравнению с производительностью накопителя на большом диске, так как запись и чтение данных производятся одновременно на несколько поверхностей дисков, входящих в состав массива. Дисковое пространство распределяется между всеми дисками массива.

Отказоустойчивость достигается либо за счет создания дополнительной копии данных, сохраняемой на другой поверхности диска, либо за счет использования контрольных сумм для выявления и исправления ошибок при восстановлении данных. Контрольные суммы также записываются на другие поверхности дисков. Выход из строя одного или большего числа дисков не приводит к потере данных, а его содержимое может быть восстановлено.

Существует несколько типов RAID-массивов. Они отличаются стоимостью, скоростью записи и чтения и надежностью сохранения информации.

RAID 0 — дисковый массив без дополнительной отказоустойчивости. Этот вариант архитектуры предусматривает простое чередо-

Рис. 12.2. Схема записи в RAID 0

вание данных при записи на несколько накопителей. Поток данных разбивается на блоки, которые записываются на диски нескольких накопителей (рис. 12.2).

За счет распределения данных, а следовательно, и операций ввода-вывода между всеми дисками массива, обеспечивается высокая производительность. Ее повышению способствует также отсутствие вычисления контрольных сумм. Однако при такой организации не обеспечивается избыточность данных, а следовательно, отказоустойчивость.

RAID 1 — дисковый массив с «зеркалированием» данных. В таком массиве предусматривается дублирование данных на «зеркальных» дисках, т. е. запись каждого блока в двух экземплярах — каждого экземпляра на свой диск. Для такой архитектуры используется один дисковый контроллер, который может выполнять одновременно две операции чтения (с двух разных дисков) и дуплексную операцию записи на зеркальные диски. Такой вариант наиболее подходит для небольших сетей, так как отличается простотой реализации, высокой скоростью записи и чтения, высокой скоростью восстановления данных из-за их 100 %-ной избыточности (данные восстанавливаются путем простого копирования с исправного диска) и возможностью получить отказоустойчивую дисковую систему всего на двух дисковых накопителях. Однако он может оказаться довольно дорогим, если требуется память большого объема. Поскольку для хранения данных используется два накопителя, то и цена такой памяти возрастает минимум в два раза.

RAID 2 — дисковый массив с использованием алгоритма Хемминга для проверки и восстановления данных. Архитектура дисковой подсистемы предусматривает запись данных на ряд дисков с добавлением кода Хемминга для обнаружения и исправления ошибок. Вся подлежащая записи информация разбивается на «слова» постоянной длины, причем длина «слова» соответствует числу дисков с данными. Здесь под «словом» понимается последова-

тельность бит постоянной длины, которые и записываются на диски с данными. Для каждого слова данных вычисляется контрольный код Хемминга, т.е. ECC (код для проверки и коррекции ошибок), также записываемый на отдельные диски для хранения контрольной информации. Этот вариант RAID отличается максимальной избыточностью, требует исключительно большого объема памяти и большого числа дисков и характеризуется максимальной стоимостью. Однако он позволяет сразу исправлять ошибки и обеспечивает высокую скорость передачи данных, увеличивающуюся с ростом числа дисков в массиве. Очень низкий коэффициент использования дискового пространства приводит к тому, что коммерческих вариантов реализации RAID 2 не существует.

RAID 3 — дисковый массив с вычислением контрольной суммы параллельно с передачей данных. Архитектура дисковой системы предусматривает запись данных с чередованием по байтам (полосам). Для записываемых на различных накопителях байтов, или полос, вычисляется значение четности (контрольной суммы), которое размещается на дополнительном накопителе.

Для реализации архитектуры RAID 3 требуется минимум два диска для хранения данных (оптимальные результаты при четырех дисках) и один диск для хранений четности. Этот способ размещения данных обеспечивает высокую скорость чтения и записи, при выходе из строя одного из дисковых накопителей общая производительность падает незначительно и довольно хорошо используется дисковое пространство. Способ применяется при передаче больших блоков данных, например файлов машинной графики.

RAID 4 — дисковый массив с независимыми дисками данных и общим диском хранения контрольной суммы. На один диск записывается весь блок данных, причем запись производится последовательно по дискам. Контрольная сумма для всех блоков одного ряда вычисляется во время операции записи данных и заносится на специальный диск для хранения контрольных сумм. При чтении производится проверка правильности записи, т.е. совпадения хранимой контрольной суммы и подсчитанной во время операции чтения.

Этот способ обеспечивает высокую скорость чтения данных и хорошо использует дисковое пространство. Однако он практически не находит применения из-за самой низкой скорости записи и сложного и неэффективного алгоритма восстановления данных при отказе одного из дисков.

RAID 5 — дисковый массив с независимыми дисками данных и равномерным распределением контрольных сумм по всем дискам. В архитектуре RAID 5 диски работают независимо друг от друга, а контрольная информация распределяется по всем дискам. Массив требует по меньшей мере двух дисков, но наиболее хорошие

Рис. 12.3. Схема записи в дисковый массив RAID 5

результаты достигаются при наличии не менее четырех. Считается, что RAID 5 целесообразнее использовать при работе с небольшими блоками данных, характерными для типичных сетевых файлов.

Большинство дисковых систем, используемых в локальных вычислительных сетях в настоящее время, строится по этому принципу. Контрольная сумма (Parity) для блоков информации одного ряда вычисляется во время операции записи и размещается на одном из накопителей (рис. 12.3), вторая сумма размещается на следующем накопителе и т. д. Во время операции чтения производится проверка правильности данных по совпадению контрольных сумм, обеспечивается высокая скорость чтения и хорошо используется дисковое пространство, однако алгоритм восстановления данных достаточно сложен; кроме того, выход из строя одного из дисковых накопителей оказывает заметное влияние на общую производительность.

Во всех RAID-системах выполняется балансирование нагрузки, т. е. использование нескольких общих источников питания так, чтобы потребляемая мощность равномерно распределялась между ними. Это делается для того, чтобы выход из строя одного источника питания не вызывал нарушения работоспособности всей системы.

При выходе из строя одного из дисков RAID часть данных теряется. При замене диска на исправный необходимо восстановить данные, т. е. «реконструировать» их. Обычно замена неисправного диска производится в режиме «горячей» замены, т. е. во время работы и обслуживания пользователей сети всеми остальными исправными дисками.

Некоторые фирмы стали комбинировать указанные уровни RAID для того, чтобы обеспечить максимальную защиту информации при конкретных ограничениях своих систем, например 53 или 50.

12.4. Построение «безотказных» систем питания

Современные компьютеры обладают высокой надежностью, однако их работоспособность зависит не только от свойств самого компьютера, но и от системы питания. Дублирование информации, средства для ее защиты, «безотказные» средства для выполнения арифметических операций и аппаратура контроля и диагностики бесполезны при отказе системы питания.

В настоящее время питание компьютеров осуществляется от сети переменного тока с помощью расположенных внутри корпуса блоков питания, т.е. блоков, преобразующих переменное напряжение сети в постоянные напряжения, подаваемые на все блоки компьютера. Блоки питания современных компьютеров служат для получения напряжений постоянного тока +3,3, +5, +12 и –12 В. В них напряжение сети переменного тока 220 В с частотой 50 Гц вначале преобразуется в напряжение с частотой 60 кГц, затем трансформируется и выпрямляется. Такая двухступенчатая схема преобразования позволяет сократить размеры понижающего трансформатора и уменьшить потери мощности, характерные для одноступенчатых схем.

При пропадании напряжения в сети питание компьютера нарушится, а результаты вычислений не будут сохранены. Для их сохранения необходимо поддерживать рабочее напряжение на схемах компьютера в течение времени, достаточного для их записи на НЖМД или другое устройство, способное хранить информацию при отсутствии питания. По этой же причине периодически записывают промежуточные результаты в энергонезависимую память, например на диск, но вычисления, проделанные в промежутке между записями, будут потеряны.

Чтобы этого не случилось, необходимо компьютер подключать не к сети питания общего пользования, а к источнику бесперебойного питания (ИБП). Задача таких источников состоит в поддержании значения выходного напряжения и частоты сети с высокой точностью. Сам ИБП подключается к сети, т.е. его входное напряжение соответствует сетевому. При нормальном напряжении сети оно передается непосредственно на выход ИБП, к которому подключен компьютер. Если оно понижается или повышается, то ИБП повышает или понижает его до номинального значения соответственно, т.е. до 220 В. Если понижение или повышение напряжения превышает некоторые пределы, обычно 165 и 280 В, то происходит переключение ИБП на работу от батарей. В состав ИБП входит аккумуляторная батарея; она обеспечивает питание компьютера в течение нескольких минут (обычно от 30 до 80 мин) после пропадания напряжения сети. Этого времени достаточно для завершения работы программ и сохранения результатов. При нормальной работе сети батарея постоянно заря-

жается специальным зарядным устройством. Обычно предусматривается возможность «горячей» замены батареи. Кроме того, в ИБП входит фильтр высокочастотных помех, что позволяет подключать к нему мощное оборудование. (Источники бесперебойного питания выпускаются различной мощности: от 500 В · А до 80 кВ · А.)

Для непрерывной работы серверов часто требуется не только наличие ИБП, возможности которых ограничены, но и создание системы безотказного питания, которая предполагает работу от нескольких сетей. Так, многие ИБП могут подключаться к двум различным сетям питания. Иногда возможность подключения к нескольким сетям предусматривается в блоке питания самого компьютера, например сервера с памятью RAID.

Контрольные вопросы

1. Какие виды отказов и сбоев характерны для современных компьютеров?
2. Как осуществляется программно-логический контроль? Какие достоинства и недостатки присущи тестовым проверкам?
3. Какой код называют корректирующим? Что такое избыточность? Как подразделяются корректирующие коды в зависимости от своего применения?
4. Каково минимальное кодовое расстояние для исправления одиночной, двойной и тройной ошибок?
5. Каким образом происходит исправление одиночной ошибки при использовании посылочного кода Хемминга?
6. Как определяется наличие ошибки при выполнении арифметических операций?
7. Какова причина появления RAID-систем? Какими возможностями обладают различные RAID-системы?
8. Что такое дисковый массив с «зеркалированием» данных? Как происходит исправление ошибки?
9. Что представляет собой RAID 2 — дисковый массив с использованием алгоритма Хемминга?
10. Какие ошибки позволяет исправить дисковый массив, построенный по принципу RAID 4? Как используется дисковая память в нем?
11. Что представляет собой дисковый массив с независимыми дисками данных и равномерным распределением контрольных сумм по всем дискам?
12. Каковы виды отказов для ЛВС? Какие средства существуют для борьбы с ними?
13. Как избежать отказов питания в вычислительной сети? К чему они могут привести?
14. Какие функции возлагают на ИБП, что входит в его состав?
15. Какие напряжения питания формируются блоками питания компьютера? Как формируются другие необходимые напряжения?

ПОСЛЕСЛОВИЕ

Компьютеры перестали быть предметом для ограниченного числа пользователей. Сегодня практически у всех студентов дома имеется персональный компьютер, который служит им для подготовки домашних заданий, обучения, игр, развлечений, доступа в Интернет. Однако, купив компьютер, многие сразу же начинают считать себя «асами» в области компьютерных дел. Они «разгоняют» процессор, увеличивают объем ОП, подключают различные устройства, загружают разные программы, игры, но при этом не знают принципов работы компьютера, его функциональных составляющих, ограничений и возможностей.

В данном учебном пособии дано представление о принципах функционирования современного компьютера. Термин «ЭВМ» не использовался, так как «электронное» будущее компьютеров туманно, а слово «машина» мало пригодно для современных персональных компьютеров. Постоянно происходящее увеличение быстродействия наталкивается на ряд принципиальных ограничений и в последнее время заметно замедлилось. Но это не значит, что так будет продолжаться вечно. Уже сегодня известно большое число технологий, которые могут заменить существующие, например твердотельная память, оптические компьютеры, плоские дисплеи различных типов и т.д. В пособии не рассматривались такие уже существующие технологии, как машинное зрение, речевой ввод-вывод и многие другие, а сделан акцент на принципах работы компьютера.

Однако не следует ждать повышения быстродействия отдельных устройств ввода-вывода, так как скорость их работы в значительной мере определяется возможностями человека по восприятию и выдаче информации.

Подключение компьютера к сети совершенно изменило облик информационной системы. Теперь благодаря Интернету стала доступной информация, которую раньше можно было получить только из книг и журналов с большой задержкой во времени. Все это приводит к резкому увеличению темпа жизни, и с этим необходимо считаться. Теперь повышение достоверности информации достигается не столько улучшением надежности отдельных устройств, сколько грамотностью человека, получающего ее из сети. Знание принципов работы компьютера, изложенных в настоящем учебном пособии, должно облегчить работу студентов.

СПИСОК ЛИТЕРАТУРЫ

1. *Бабич Н. П.* Компьютерная схемотехника. Методы построения и проектирования / Н. П. Бабич. — Киев : МК-Пресс, 2004.

2. *Бродин В. Б.* Микропроцессор i486. Архитектура, программирование, интерфейс / В. Б. Бродин, И. И. Шагурин. — М. : Диалог-МИФИ, 1993.

3. *Водяхо А. И.* Высокопроизводительные системы обработки данных / А. И. Водяхо, Н. Н. Горнец, Д. В. Пузанков. — М. : Высш. шк., 1997.

4. *Воеводин В. В.* Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. — СПб. : БХВ-Петербург, 2002.

5. *Горнец Н. Н.* Архитектура современных ЭВМ : учеб. пособие / Н. Н. Горнец. — М. : МГТУ ГА, 2000.

6. *Гук М.* Аппаратные средства IBM PC : энциклопедия / М. Гук. — СПб. : Питер, 1999.

7. *Каган Б. М.* Электронные вычислительные машины и системы / Б. М. Каган. — М. : Энергоатомиздат, 1991.

8. *Корнеев В. В.* Современные микропроцессоры / В. В. Корнеев, А. В. Киселев. — 2-е изд. — М. : НОЛИДЖ, 2000.

9. *Кулаков В. Г.* Программирование дисковых подсистем / В. Г. Кулаков. — СПб. : Питер, 2002.

10. *Ларионов А. М.* Периферийные устройства в вычислительных системах / А. М. Ларионов, Н. Н. Горнец. — М. : Высш. шк., 1991.

11. *Ларионов А. М.* Вычислительные комплексы, системы и сети / А. М. Ларионов, С. А. Майоров, Г. И. Новиков. — Л. : Энергоатомиздат, 1987.

12. *Лысиков Б. Г.* Цифровая и вычислительная техника : учебник / Б. Г. Лысиков. — Минск : УП «Экоперспектива», 2002.

13. *Мураховский В. И.* Устройство компьютера / В. И. Мураховский ; под ред. С. В. Симоновича. — М. : АСТ-Пресс книга, 2004.

14. *Олифер В. Г.* Компьютерные сети / В. Г. Олифер, Н. А. Олифер. — СПб. : Питер, 1999.

15. *Олифер В. Г.* Сетевые операционные системы / В. Г. Олифер, Н. А. Олифер. — СПб. : Питер, 2001.

16. *Половов Р. М.* Бортовые цифровые вычислительные устройства и машины : учеб. пособие. Ч. I и II / Р. М. Половов, А. Г. Рощин. — М. : МГТУ ГА, 2003, 2004.

17. Системное программное обеспечение: файловые системы ОС Unix и Windows NT / [И. В. Машечкин, М. И. Петровский, П. Д. Скулачев и др.]. — М. : Диалог-Москва, 1997.

18. *Столлингс У.* Структурная организация и архитектура компьютерных систем / У. Столлингс. — 5-е изд. — М. : Вильямс, 2002.

20. *Таненбаум Э.* Архитектура компьютера / Э. Таненбаум. — СПб. : Питер, 2002.

21. *Таненбаум Э.* Современные операционные системы / Э. Таненбаум. — СПб. : Питер, 2002.

22. *Таненбаум Э.* Компьютерные сети / Э. Таненбаум. — СПб. : Питер, 2004.

23. *Тейлор Б.* RAID как средство спасения данных / Б. Тейлор // PC Magazine/Russian Edition, спец. выпуск, 1993—1994.

24. *Уолренд Дж.* Телекоммуникационные и компьютерные сети / Дж. Уолренд. — М. : Постмаркет, 2001.

25. *Цилькер Б. Я.* Организация ЭВМ и систем / Б. Я. Цилькер, С. А. Орлов. — СПб. : Питер, 2004.

ОГЛАВЛЕНИЕ

Учебное издание

Горнец Николай Николаевич
Рощин Алексей Григорьевич
Соломенцев Виктор Владимирович

Организация ЭВМ и систем

Учебное пособие

Редактор *Л. В. Толочкова*
Технический редактор *Е. Ф. Коржуева*
Компьютерная верстка: *Л. М. Беляева*
Корректоры *И. В. Могилевец, И. Н. Волкова*

Изд. № А-1551-I. Подписано в печать 14.12.2005. Формат 60×90/16.
Гарнитура «Таймс». Печать офсетная. Бумага тип. № 2. Усл. печ. л. 20,0.
Тираж 4000 экз. Заказ №15859.

Издательский центр «Академия». www.academia-moscow.ru
Санитарно-эпидемиологическое заключение № 77.99.02.953.Д.0047963.07.04 от 20.07.2004.
117342, Москва, ул. Бутлерова, 17-Б, к. 360. Тел./факс: (495)330-1092, 334-8337.

Отпечатано на Саратовском полиграфическом комбинате.
410004, г. Саратов, ул. Чернышевского, 59.

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

ISBN 5-7695-2269-0

Издательский центр «Академия»
www.academia-moscow.ru

Высшее профессиональное образование

Н. Н. Горнец
А. Г. Рощин
В. В. Соломенцев

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

Учебное пособие

Информатика и вычислительная техника

ACADEMIA