

МИКРОЭВМ, МИКРОПРОЦЕССОРЫ И ОСНОВЫ ПРОГРАММИРОВАНИЯ

Под общей редакцией
доктора технических наук
А. Н. Морозевича

Допущено Министерством
народного образования БССР
в качестве учебного пособия для учащихся
средних специальных учебных заведений
по специальности 1202 «Производство станков
с программным управлением и роботов»

Минск
«Вышэйшая школа»
1990

ББК 32.973я723

М59

УДК 681.3:621.382.049.721.14 + 681.3.06] (075.32)

Авторы: А. Н. Морозевич, А. Н. Дмитриев, В. Н. Мухаметов, В. В. Панькова, А. П. Пашкевич, В. Н. Путков, Г. В. Таранов

Рецензенты: ректор Ленинградского института точной механики и оптики, профессор, доктор технических наук *Г. И. Новиков*; цикловая комиссия ЭВМ Минского радиотехнического техникума

МикроЭВМ, микропроцессоры и основы программирования: Учеб. пособие / А. Н. Морозевич, А. Н. Дмитриев, В. Н. Мухаметов и др.; Под общ. ред. А. Н. Морозевича. — Мн.: Выш. шк., 1990. — 352 с.: ил.

ISBN 5-339-00458-9.

Рассматриваются арифметические, логические и схемотехнические основы ЭВМ и микроЭВМ, вопросы организации микропроцессоров и микроЭВМ, их периферийных устройств, основы программирования, а также применение микроЭВМ в робототехнике и системах управления.

Для учащихся средних специальных учебных заведений по специальности «Производство станков с программным управлением и роботов». Может быть полезно студентам вузов, а также специалистам, занимающимся разработкой микропроцессорных устройств.

2404000000—109
М 78—90
М304(03) — 90

ББК 32.973я723

ISBN 5-339-00458-9

© Коллектив авторов, 1990

ПРЕДИСЛОВИЕ

С точки зрения большинства потребителей микроЭВМ можно определить как ЭВМ, построенную на основе электронных схем высокой степени интеграции, изготавливаемую в виде промежуточного продукта (полуфабриката) и доукомплектовываемую пользователем теми аппаратными и (или) программными средствами, которые определены им для каждого конкретного случая. Такие машины становятся обычными, доступными для потребителя функциональными устройствами, направления применения которых все чаще определяются специалистами в соответствующих областях науки и техники (а не разработчиками ЭВМ, как это было несколько лет назад). Поэтому принципы организации и функционирования микроЭВМ, характеристики и возможности их использования являются основным элементом знаний инженеров и техников, занимающихся разработкой и эксплуатацией микропроцессорных систем управления.

Данная книга является учебным пособием по курсу «МикроЭВМ, микропроцессоры и основы их программирования» для учащихся техникумов, обучающихся по специальности «Производство станков с программным управлением и роботов». Используя опыт преподавания указанного курса в Минском политехникуме и аналогичных курсов в Минском радиотехническом институте и Академии наук Белорусской ССР, авторы стремились осветить в доступной форме, ясно и логично арифметические, логические и схемотехнические основы ЭВМ, принципы построения и особенности работы и использования микропроцессоров и микроЭВМ. Книга может быть полезна студентам вузов при изучении таких курсов, как «Вычислительные устройства промышленных роботов» (специальность «Эксплуатация промышленных роботов»), «Основы электроники и вычислительной техники» (специальность «Прикладная математика»), и других, на-

правленных на освоение микропроцессорных средств и систем управления, а также специалистам, занимающимся разработкой и эксплуатацией различного рода микропроцессорных устройств и систем, но не имеющим для этих целей необходимого базового образования.

Глава 1, § 8.6 написаны А. Н. Морозевичем; гл. 2 — совместно А. Н. Морозевичем и В. В. Паньковой; гл. 3 — В. Н. Путковым; гл. 4 — Г. В. Тарановым; гл. 5 — В. Н. Мухаметовым; гл. 6 и 7 (кроме § 7.2) — А. Н. Дмитриевым; § 7.2 — совместно А. Н. Дмитриевым и В. Н. Мухаметовым; § 8.1—8.5 — А. П. Пашкевичем; § 8.7 А. Н. Морозевичем, А. А. Денисовым и Б. Б. Трибуховским совместно.

Авторы выражают глубокую признательность рецензентам — цикловой комиссии ЭВМ Минского радиотехнического техникума и доктору технических наук, профессору, ректору Ленинградского института точной механики и оптики Г. И. Новикову за ценные замечания, способствовавшие улучшению содержания книги.

Все пожелания и замечания по книге просим присылать по адресу: 220048, Минск, проспект Машерова, 11, издательство «Вышэйшая школа».

Авторы

Глава 1. ПРИНЦИПЫ ОРГАНИЗАЦИИ ЭВМ

1.1. История развития ЭВМ

Когда говорят о техническом прогрессе в области электронных вычислительных машин, то обычно выделяют пять этапов, которые рассматривают во взаимосвязи с применяемой на каждом из них элементной базой: *электронные лампы, полупроводниковые* (дискретные) диоды и *транзисторы, интегральные микросхемы различной степени интеграции* (рис. 1.1).

Первые цифровые электронные **вычислительные** машины (ЭВМ), изготовленные с использованием электронных ламп (1-е поколение ЭВМ), были созданы исключительно для выполнения объемных научно-технических расчетов. Эти установки имели гигантские по сегодняшним масштабам размеры, отличались большим энергопотреблением, требовали высоких капитальных и эксплуатационных расходов. Например, первая в мире ЭВМ «ЭНИАК» созданная в 1945 г. учеными Пенсильванского университета (США), весила 30 т, содержала 18 000 электронных ламп и стоила почти 2,8 млн долларов по ценам того времени. При этом она выполняла около 5000 операций сложения или примерно 360 операций умножения в секунду.

Первые отечественные ламповые вычислительные машины МЭСМ и БЭСМ были созданы под руководством академика С. А. Лебедева. МЭСМ (малая электронная счетная машина), созданная в 1951 г., сыграла важную роль в подготовке первых в стране программистов, инженеров и конструкторов ЭВМ, интенсифицировала разработку электронных элементов специально для применения в ЭВМ. БЭСМ (большая электронная счетная машина), являясь в то время самой быстродействующей ЭВМ в мире (8000 опер/с), открыла серию машин, получивших широкое распространение в СССР. В первой половине 50-х гг. у нас в стране появились ЭВМ серий «Стрела» и «Урал», а в 60-х гг. — «Проминь», «Мир», «Минск», «Раздан». Эти машины могли справиться с

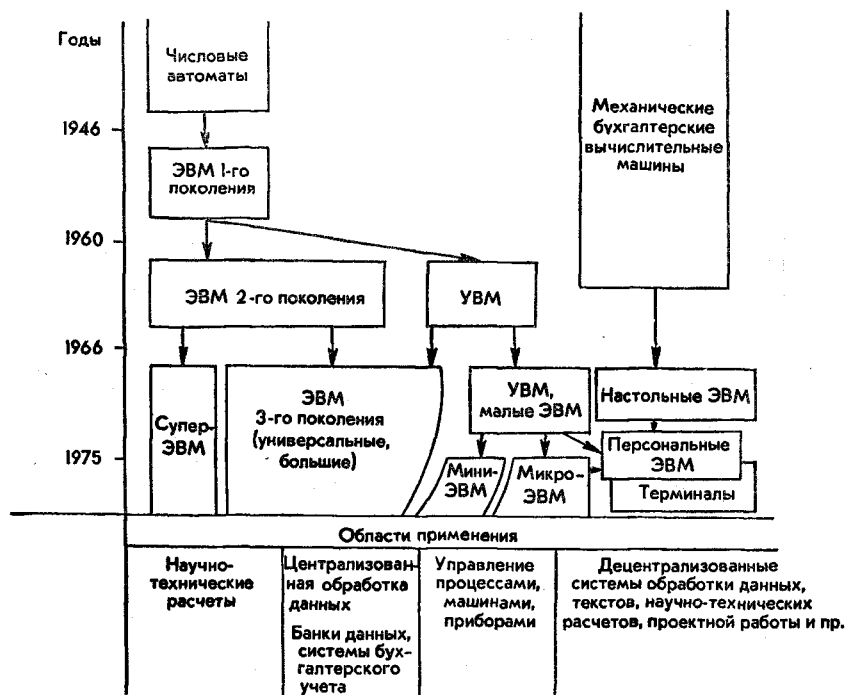


Рис. 1.1

широким кругом математических и логических задач, встречающихся при решении научных и сложных инженерных проблем.

Освоение и промышленный выпуск полупроводниковых приборов обеспечили замену «громоздких и горячих» электронных ламп «миниатюрными и теплыми» транзисторами. Это привело к созданию вычислительных устройств, характеризующихся более высокими быстродействием, надежностью и функциональными возможностями при меньших габаритах, стоимости и эксплуатационных расходах (2-е поколение).

Однако представить смену поколений ЭВМ лишь как замену элементной базы, приведшей к повышению технических характеристик, было бы неверно. Новые элементы преобразили «душу» ЭВМ. К этому времени она научилась «понимать» соответствующий язык, и любой человек, владеющий этим языком, мог «общаться» с машиной. Однако общение это осуществлялось, как правило, посред-

ством операторов, обслуживающих устройства подготовки данных (перфораторы) и ЭВМ. В непосредственный контакт с машиной вступал лишь «привилегированный класс» операторов-программистов и инженеров по эксплуатации ЭВМ.

Серийные машины 2-го поколения «Минск-32» и «Урал-16» имели быстродействие порядка 250 000 и 100 000 опер/с. Их оперативная память удерживала соответственно 65 000 и 500 000 чисел. ЭВМ «Минск-32», например, могла работать со 136 внешними устройствами, а управлял ею один оператор с помощью пишущей машинки.

Еще более совершенной была БЭСМ-6 (выпуск 1967 г.). Ее быстродействие — 1 млн опер/с. Оперативная память машины позволяла хранить 128 000 чисел, а промежуточная на магнитном барабане — 512 000 чисел. Каждый из 32 подключаемых к ЭВМ магнитофонов обеспечивал хранение на магнитной ленте до миллиона машинных слов (5000 страниц текста). БЭСМ-6 отличается не только то, что она была одной из самых лучших в мире машин второго поколения, но и удивительная «живучесть», обеспечившая ее эксплуатацию до настоящего времени.

Появление быстродействующих устройств ввода (способных пропускать до 1000 перфокарт в минуту), алфавитно-цифровых печатающих устройств (АЦПУ), графопостроителей дало возможность гибко менять форму выдачи результатов, например печатать их в виде таблиц со словесным описанием приведенных величин либо оформлять в виде готовых графиков. Все это существенно облегчило обработку результатов, повысило производительность человеческого труда. При этом возникло понятие «машина для обработки данных». В отличие от ЭВМ для научно-технических расчетов эта машина должна обладать такими свойствами, как хранение (накопление, запоминание), ввод и вывод больших массивов чисел, тогда как процессы обработки (вычислительные операции) отступают на задний план.

На втором этапе развития ЭВМ были предприняты попытки использовать вычислительную машину для управления промышленными технологическими процессами, породившие так называемые *управляющие вычислительные машины* (УВМ). Такие ЭВМ в первую очередь наблюдали за измеряемыми показателями процессов, рассчитывали и вырабатывали управляющие воздействия либо

помогали (что более характерно для ЭВМ второго поколения) оператору вести управление. При этом возникла новая для ЭВМ ситуация: результаты расчетов могли быть использованы лишь тогда, когда они не только верны, но и своевременно подготовлены для использования. Такой режим работы ЭВМ специалисты называют работой в *реальном масштабе времени*.

К концу 60-х гг. стало ясно, что для повышения эффективности использования ЭВМ при обработке данных и управлении необходимо создавать модели ЭВМ разной производительности, но одинаковые по своей организации и обладающие программной совместимостью. Последнее означает возможность использовать запас программ, написанных для одной ЭВМ, на машинах других моделей, за счет чего снижаются затраты на обработку информации.

Принцип программной совместимости и технология интегральных схем положили начало третьему этапу развития ЭВМ. Для машин 3-го поколения характерно не только улучшение габаритно-стоимостных показателей, но и модульный принцип организации технических и программных средств, обеспечивший возможность составлять приспособленную для соответствующего конкретного назначения конфигурацию ЭВМ. Машины 3-го поколения обрабатывают не только числа, но и слова, тексты, т. е. оперируют буквенно-цифровой информацией. Изменилась и форма общения человека с машиной. Пользователи получили доступ к ЭВМ. Машина через выносной терминал «сама пришла» к человеку в его служебное помещение. Спираль развития вычислительной техники и ее использования человеком завершила очередной виток.

Начало создания универсальных машин третьего поколения положила фирма IBM (США), приступившая в 1966 г. к выпуску машин серии IBM-360. Выпуск машин данного класса, совместимых с IBM, в рамках единой системы ЭВМ (ЕС ЭВМ) в странах — членах СЭВ начался в 1972 г.

В ЕС ЭВМ приняты единые стандарты на технические характеристики всех устройств и узлов, на системы кодов, операций, средств программирования. Все модели ЕС ЭВМ имеют общий состав периферийных устройств, обеспечивающих ввод-вывод информации. В них предусмотрена возможность связи с абонентами по телефонно-телеграфным линиям с использованием терминальных пультов, включающих устройства алфавитно-цифрового и графического

ческого отображения данных на экранах электронно-лучевых трубок. Каждая модель ЕС ЭВМ имеет свой собственный процессор, являющийся как бы ядром этой модели. Весь ряд таких моделей строится в порядке возрастания их быстродействия от нескольких тысяч (ЕС 1010, Венгрия) до миллионов (ЕС 1065, СССР) операций в секунду.

На третьем этапе линия управляющих ЭВМ частично снова сливается с линией ЭВМ для обработки данных (см. рис. 1.1). Однако отдельные ветви УВМ продолжают развиваться самостоятельно. Появились меньшие по объему установки. При этом возникли новые понятия: малые ЭВМ, малые управляющие ЭВМ, мини-ЭВМ. В 1974 г. страны — члены СЭВ, а также Куба и Румыния объединили свои усилия в области создания семейства малых ЭВМ (СМ ЭВМ), предназначенных для использования в информационно-измерительных и управляющих системах. С появлением малых ЭВМ возникло еще одно направление использования вычислительной техники: децентрализованная обработка данных и использование ЭВМ в непосредственной близости от рабочих мест (настольные ЭВМ).

На этом же этапе зародились суперЭВМ, целевой установкой при разработке которых было и остается достижение максимальной производительности вычислительных процессов (нескольких сотен миллионов операций в секунду). Их возникновение определено необходимостью решения научно-технических задач (например, современных задач аэродинамики и ядерной физики), предполагающих выполнение значительного числа операций (для указанного примера не менее 10^{13}) за ограниченный промежуток времени. Очевидно, что суперЭВМ весьма сложны и дороги, а поэтому в настоящее время насчитывается немногим более 150 таких машин во всем мире.

Четвертое поколение ЭВМ служит еще одним примером перехода количества в качество. При их создании как будто не произошло ничего особенного. Просто интеграция электронных схем повысилась настолько, что стало технически возможным сосредоточить значительное число функциональных устройств в одной большой интегральной схеме (БИС) и, таким образом, изготовить по этой технологии большие (по функциональным возможностям) блоки или всю ЭВМ в целом. Но появление БИС — это не только создание более совершенной элементной базы ЭВМ. Оно создало предпосылки для качественного изменения

вычислительной техники. Применение БИС привело к новым представлениям о функциональных возможностях элементов и узлов ЭВМ. Разработка (1969 г., Intel, США) и промышленное освоение микропроцессоров (МП) обеспечили широкие возможности для децентрализации вычислительной мощности и встраивания вычислительных средств в оборудование и приборы.

До этого соотношение стоимости и производительности было в пользу больших вычислительных установок и потому господствовали тенденции возрастающих централизации и мощности ЭВМ. С появлением МП стоимость ЭВМ резко снизилась, что послужило толчком к развитию децентрализованного принципа построения вычислительных систем. Один из примеров — «Машина связи» (Thinking Machines, США). Эта машина с объемом $1,5 \text{ м}^3$ может производить несколько миллиардов операций в секунду, т. е. превосходит в скорости гораздо более крупные суперЭВМ, будучи в четыре раза дешевле их. При одной из демонстраций «Машина связи» за одну двадцатую долю секунды «прочитала» 16 000 сообщений типа газетных новостей и за 3 мин рассчитала схему кристалла с 4000 транзисторов. Столь высокое быстродействие «Машины связи» объясняется тем, что она содержит более 65 000 МП, каждый из которых обладает собственной памятью небольшого объема. Более того, каждый МП прямо или косвенно соединен со всеми остальными узлами, так что схема машины может перестраиваться электронным путем в соответствии с особенностями задачи, которую в данный момент предстоит решить.

На базе МП строятся микроЭВМ и микроконтроллеры. МикроЭВМ содержит МП вместе с запоминающим устройством, устройством ввода-вывода информации и устройствами связи. Эти устройства могут выполняться в виде отдельных БИС либо на одном кристалле с процессором (однокристалльные микроЭВМ).

Если МП выполняет функции управления, его называют контроллером (нельзя считать контроллер контролирующим устройством — это не контролёр, а устройство управления). В современных ЭВМ микропроцессоры управляют, например, работой внешних устройств: дисковой и магнитофонной памятью, печатающим устройством, графопостроителем и т. д.

Эволюция микропроцессорной техники 70-х гг. — МП, микроЭВМ, персональные ЭВМ — в основном напоминает пройденные в 60-е гг. этапы развития мини-ЭВМ: от

встраиваемых контроллеров — к функциям универсальных ЭВМ в системах распределенной обработки данных. Однако впечатляет разница в масштабах: общий парк мини-ЭВМ составлял 200 тыс. экземпляров за первые 10 лет их производства, тогда как общий объем производства МП оценивался к 1984 г. на уровне 200 млн в год. Мини-ЭВМ приобреталась для работы, если в ней были заинтересованы 10—30 сотрудников (например, исследовательская группа или лаборатория, технологический участок, небольшая контора и т. д.), а универсальная микроЭВМ по экономическим соображениям стала индивидуальным инструментом, так называемой *персональной ЭВМ*.

С середины 70-х гг. активно прорабатываются основы для построения машин 5-го поколения. В настоящее время еще рано говорить о завершении этих работ, хотя уже подготовлен теоретический и технический базис, позволяющий создавать новую архитектуру и обеспечивать реализацию новых функций, направленных на интеллектуализацию ЭВМ. Этот базис — развивающаяся технология сверхбольших интегральных схем (СБИС), создание памяти повышенного объема, возрастающие возможности высокоскоростных элементов, расширение исследований в области искусственного интеллекта и распознавания образов, а также совместное развитие коммуникационных систем и систем обработки информации.

Если рассматривать историю вычислительной техники с точки зрения развития информационной технологии, то в ней можно выделить три этапа, кратко характеризующиеся следующим образом.

Первый этап (50—60-е гг.) — экономия машинных ресурсов. Машин мало, нерешенных задач счетного характера множество. Основная из них: экономия времени решения при ограниченном объеме памяти. Для ее выполнения обеспечивалась такая организация вычислительного процесса, при которой максимально загружался процессор (самая дорогостоящая часть ЭВМ того времени). Чтобы ускорить процесс кодирования (подготовки задач к решению), были созданы алгоритмические языки АЛГОЛ, ФОРТРАН и др.

Второй этап (середина 60-х — начало 80-х гг.) — экономия человеческих ресурсов. Успехи развития микроэлектроники привели к быстрому снижению удельной стоимости машинной операции и единицы объема оперативной памяти, тогда как расходы на разработку и сопровождение программ не снижались, а в ряде случаев

имели тенденцию к росту. На этом этапе (т. е. через десять лет после первых успешных попыток подчинить ресурсы ЭВМ задаче автоматизации программирования: созданию трансляторов) экономия человеческих, а не машинных ресурсов стала, наконец, центральной проблемой технологии программирования. От технологии эффективного использования программ к технологии эффективного программирования — так можно определить общее направление смены критериев эффективности на первом и втором этапах.

Третий этап (от начала 80-х гг. до настоящего времени) — формализация знаний. До середины 70-х гг. с ЭВМ работали в среднем один или несколько профессиональных программистов, задачей которых было программирование формализованных знаний. Но за 30 лет развития вычислительной техники заметная часть того задела ранее формализованных знаний, который был накоплен человечеством за последние 300 лет интенсивного развития точных наук, оказалась записанной в машинных программах. К концу 1983 г. в подавляющем большинстве случаев (9 из 10) за пультом ЭВМ находился не программист, а так называемый «непрограммирующий профессионал», профессионально владеющий «тайнами ремесла» в конкретной предметной области, где может быть полезна ЭВМ, но не имеющий профессиональной подготовки в области вычислительной техники и программирования. Самостоятельная формализация (автоформализация) профессиональных знаний уже на первом ее этапе (автоматизация рутинной работы, выполняемой специалистами) гарантирует огромный народнохозяйственный эффект. Но для этого должна быть обеспечена «дружеская» реакция машины на любые, в том числе неадекватные, действия пользователя ЭВМ.

1.2. Принцип программного управления

Название «электронная вычислительная машина» соответствует изначальной области применения ЭВМ — выполнению научно-технических расчетов. Однако для современных ЭВМ больше соответствует определение *программно управляемая искусственная (инженерная) система, предназначенная для восприятия, хранения, обработки и передачи информации.*

Такое определение подчеркивает, что в основу ЭВМ положен принцип программного управления. Один из спо-

совов его реализации был предложен в 1945 г. американским математиком Дж. фон Нейманом, и с тех пор неймановский принцип программного управления используется в качестве основного принципа построения ЭВМ. Этот принцип состоит в следующем:

информация кодируется в двоичной форме и разделяется на единицы (элементы) информации — слова;

разнотипные слова информации различаются по способу использования, но не способами кодирования;

слова информации размещаются в ячейках памяти машины и идентифицируются номерами ячеек, которые называются адресами слов;

алгоритм представляется в форме последовательности управляющих слов — команд, которые определяют наименование операции и слова информации, участвующие в операции. Алгоритм, представленный в терминах машинных команд, называется программой;

выполнение вычислений, предписанных алгоритмом, сводится к последовательному выполнению команд в порядке, однозначно определяемом программой. Первой выполняется команда, заданная пусковым адресом программы. Обычно это адрес первой команды программы. Адрес следующей команды однозначно определяется в процессе выполнения текущей команды и может быть либо адресом следующей по порядку команды, либо адресом любой другой команды. Процесс вычислений продолжается до тех пор, пока не будет выполнена команда, предписывающая прекращение вычислений.

Перечисленные слагаемые неймановского принципа подробно рассмотрены в последующих главах учебного пособия. Здесь же подчеркнем лишь то, что именно программа «настраивает» ЭВМ на получение требуемых результатов. Замена программы приводит к изменению функций, реализуемых ЭВМ.

Неймановский принцип программного управления не лишен недостатков. Во-первых, представление информации в двоичной форме (нетрадиционной для человека) существенно затрудняет «общение» человека с машиной. ЭВМ с развитой системой интерпретации (принцип разработан академиком В. М. Глушковым) обеспечивают восприятие алгоритмов, записанных на языках высокого уровня — в виде знаков операций, наименований величин и данных, представляемых в естественной форме, причем указанные возможности реализуются за счет введения в ЭВМ нетрадиционных средств адресации и операций над

информацией. Во-вторых, неймановский принцип предполагает, что коды слов информации не зависят от типа информации. Это приводит к тому, что программист сам обязан следить за тем, чтобы для обработки информации определенного типа, например целых или действительных чисел, использовались соответствующие операции, чтобы был запрограммирован перевод чисел из одной формы представления в другую и пр. Если эти правила не соблюдаются, то в программе появляются ошибки, а результат может получиться непредсказуемым. Английский ученый Дж. Айлиф предложил отображать тип информации (числа, адреса, команды) в кодах данных. В результате операция, указываемая в команде, производится машиной в форме, соответствующей типу информации. Это приводит к сокращению списка машинных команд (например, достаточно иметь команду «сложить»; машина же сама «разберется», как складывать: по правилу сложения целых чисел или по правилу сложения действительных чисел) и уменьшению числа ошибок в программе. В-третьих, память неймановской машины сугубо линейна, так как идентифицируется последовательностью адресов, например от 0 до M . И какой бы ни была структура данных, т. е. из каких бы элементов (скаляров, векторов, матриц) ни состояли данные и как бы они ни были взаимосвязаны, программист должен эти данные спроецировать на линейную цепочку адресов 0, 1, ..., M . Затем при составлении программы ему приходится определять способ выделения адресов, соответствующих отдельным структурным элементам данных. Процедуры размещения информации в памяти и выделения элементов информации оказываются весьма сложными. Для упрощения процесса программирования и самой программы Дж. Айлиф предложил вносить описание структуры информации непосредственно в память машины, за счет чего обеспечивается возможность автоматического выявления адресов отдельных элементов в процессе работы машины. Ясно, что дополнительные возможности ЭВМ должны обеспечиваться за счет введения в машину дополнительной аппаратуры.

До последнего времени попытки определить наиболее рациональные принципы построения ЭВМ обычно заканчивались созданием машин, построенных на неймановском принципе, наращивание возможностей обеспечивалось преимущественно программными средствами. Это объясняется, видимо, тем, что возможности неймановских

машин обеспечивают потребности человека во многих приложениях его деятельности. Однако к настоящему времени появились такие сложные задачи, затраты на программирование и решение которых на ЭВМ экономически и технически нецелесообразны (либо невозможны). В связи с этим ощущается потребность в пересмотре классического неймановского принципа построения ЭВМ с тем, чтобы приблизить машинные формы представления данных и алгоритмов к естественным, повседневно используемым способам представления и обработки информации.

1.3. Обобщенная структура ЭВМ

Определение электронной вычислительной машины (с учетом программного принципа управления) предполагает, что ЭВМ строится по следующей схеме (рис. 1.2).

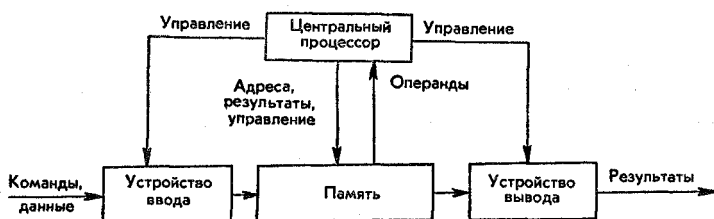


Рис. 1.2

Устройства ввода и вывода обеспечивают ЭВМ связь с внешним миром. Устройства ввода осуществляют считывание информации (программ и данных) с определенных носителей (перфокарт, перфолент, клавиатур и т. п.) и представление считанной информации в форме электрических сигналов, воспринимаемых памятью. Устройства вывода преобразуют электрические сигналы, поступающие из памяти и несущие в себе информацию о результатах обработки данных в форме печатного текста, графиков, изображений на экранах видеоконтрольных устройств, перфорационных отверстий на соответствующих картах или лентах и т. п. В отдельных случаях ввод и вывод реализуются одним физическим устройством, например электрической пишущей машинкой, с клавиатуры которой вводится, а на лист бумаги выводится информация. Такие устройства, обеспе-

чивающие как ввод, так и вывод информации, называются устройствами ввода-вывода. Этим термином часто пользуются для обозначения любого устройства, относящегося к классу устройств ввода и вывода информации.

Память ЭВМ содержит программы, исходные данные, промежуточные результаты и другую информацию, необходимую ЭВМ для совершения различных (требуемых) операций и для взаимной связи между отдельными частями устройства. В большинстве ЭВМ имеется несколько различных устройств для хранения информации. В основной (достаточно быстродействующей) памяти хранится оперативная информация. Поэтому основную память часто называют *оперативной*. Редко используемая информация может храниться вне ЭВМ, например на магнитных лентах, магнитных или оптических дисках.

Вычисления, заданные программой, реализуются центральным процессором, или (когда термин не вызывает неопределенности) просто процессором. Его функцией является выборка команд из памяти и выполнение действий, предписанных ими. Центральный процессор выполняет все основные операции, за исключением операций ввода-вывода информации. Команды ввода-вывода, как и любые другие, поступают в центральный процессор, который выступает инициатором операций ввода-вывода, а сами операции реализуются устройствами ввода-вывода. Следует подчеркнуть, что, «воспринимая» программу, центральный процессор управляет работой всех устройств ЭВМ.

ЭВМ работает следующим образом. Программа и исходные данные, представленные на машинном носителе информации, например на магнитных дисках или лентах, реже на перфокартах или перфолентах, считываются соответствующим устройством ввода и загружаются в память ЭВМ. Затем в центральный процессор с пульта управления (или клавиатуры дисплея) посылается пусковой адрес программы и команда на выполнение программы. Центральный процессор начинает выполнять программу от команды с заданным адресом, что сводится к последовательной выборке команд из памяти и их выполнению средствами процессора и устройств ввода-вывода. Этот процесс заканчивается в момент выборки команды остановки или окончания обработки информации.

Перечисленные выше компоненты имеет любая ЭВМ, как бы и кем бы она ни была создана. Современные ЭВМ

решают задачи из разных областей человеческой деятельности. Как ни парадоксально, но именно многообразие потребностей привело к однообразной обобщенной структуре машин. Причем эта структура сегодня наилучшим образом в среднем решает все задачи. Это значит, что для решения каждой задачи можно было бы придумать свою структуру ЭВМ, которая могла быть проще либо дешевле и надежнее, а также быстрее решать эту задачу. Но «стандартная» (неймановская) структура остается оптимальной для решения множества задач, хотя для каждой в отдельности она далека от совершенства.

Конечно, такая «оптимальная в среднем» схема весьма расточительна, так как для конкретной ситуации она не является оптимальной. Эти соображения заставляют разработчиков создавать гамму вычислительных средств: от специализированных ЭВМ до вычислительных систем и сетей.

Контрольные вопросы и задания

1. Когда появилась первая в мире ЭВМ?
2. Когда и под чьим руководством была создана первая ЭВМ в СССР?
3. Перечислите основные признаки ЭВМ 1, 2, 3, 4 и 5-го поколений.
4. Что такое принцип программного управления?
5. Изобразите обобщенную структуру ЭВМ.

Глава 2. АРИФМЕТИЧЕСКИЕ ОСНОВЫ ЭВМ

2.1. Позиционные системы счисления

Система счисления — это совокупность цифровых знаков и правил их записи, применяемая для однозначной записи чисел. Все системы счисления подразделяются на позиционные и непозиционные.

Непозиционной называется такая система счисления, в которой значение цифры не зависит от ее положения в ряду цифр, изображающих число. Примером является римская система счисления, в которой для обозначения отдельных чисел используются буквы римского алфавита. Цифры в римской системе обозначаются различными знаками:

I — I; 3 — III; 5 — V; 10 — X; 50 — L; 100 — C; 500 — D; 1000 — M. Запись числа осуществляется по правилу: каждый меньший знак, поставленный справа от большего, прибавляется к его значению, а слева — вычитается из него: так, XC — 90; CX — 110; MCMLXXXVIII — 1988. Выполнять арифметические действия в непозиционных системах неудобно. Поэтому в настоящее время эти системы не используются для расчетов.

Позиционной называется такая система счисления, в которой значение цифры зависит от ее положения в ряду цифр, изображающих число, т. е. *веса*. В десятичной системе счисления вес каждой последующей цифры в 10 раз больше веса предыдущей. Например, цифра 2 в числе 1235 имеет значение 200, так как она расположена в третьей справа позиции числа.

Позиционная система счисления (ПСС) характеризуется количеством различных цифр, используемых для записи чисел. Максимальное количество различных цифр, используемых для записи чисел в данной системе счисления, называется *основанием* системы счисления.

Любое число, записанное в p -ичной ПСС, может быть представлено в следующем виде:

$$X_{(p)} = a_n p^n + a_{n-1} p^{n-1} + \dots + a_0 p^0 + a_{-1} p^{-1} + \dots + a_{-m} p^{-m}, \quad (2.1)$$

где a_n, \dots, a_{-m} — любая цифра (символ), используемая в данной ПСС из множества $\{0, 1, \dots, p-1\}$; n, \dots, m — номера разрядов числа; p — основание ПСС, которым может быть любое целое число, кроме $|p| \leq 1$ и $|p| = \infty$; p^n, \dots, p^{-m} — веса разрядов. Например, число $X_{(10)} = 1235,87$ по формуле (2.1) имеет вид:

$$X_{(10)} = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 5 \cdot 10^0 + 8 \cdot 10^{-1} + 7 \cdot 10^{-2}.$$

В ЭВМ применяют ПСС с недесятичным основанием: двоичную, восьмеричную, шестнадцатеричную и др. В табл. 2.1 показано соответствие записи чисел в десятичной, восьмеричной и шестнадцатеричной системах счисления.

Двоичная ПСС получила самое широкое применение в ЭВМ благодаря следующим достоинствам.

Таблица 2.1

$X_{(10)}$	$X_{(8)}$	$X_{(8)}$	$X_{(16)}$
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1. Числовая информация в ЭВМ отождествляется с состоянием используемых двоичных физических элементов. В двоичной ПСС $a_i \in \{0, 1\}$. Поэтому для физического представления a_i достаточно использования элементов с двумя устойчивыми состояниями, кодируемыми 1 и 0. Например, транзистор может быть в открытом или закрытом состоянии, а следовательно, иметь на выходе высокое или низкое напряжение, ферритовый сердечник в устойчивом состоянии может иметь положительную или отри-

цательную остаточную магнитную индукцию, лампочка включена или выключена, отверстия на перфокарте пробиты или нет. Такие элементы принято называть двухпозиционными или двоичными. Очевидно, что реализация элементов, которые должны различать одно из двух состояний (0 или 1), оказывается проще и надежнее, чем реализация элементов, которые должны различать одно из 10 состояний.

2. Арифметические операции выполняются наиболее просто. Например, таблицы сложения и умножения одноразрядных двоичных чисел имеют соответственно вид:

$$\begin{array}{ll} 0 + 0 = 0; & 0 \times 0 = 0; \\ 0 + 1 = 1; & 0 \times 1 = 0; \\ 1 + 0 = 1; & 1 \times 0 = 0; \\ 1 + 1 = 10; & 1 \times 1 = 1. \end{array}$$

3. Процесс синтеза схем ЭВМ упрощен, так как обозначение переменных и функций в используемом математическом аппарате алгебры логики, принимающих два значения 0 или 1, совпадает с двоичными цифрами.

В то же время громоздкость записи чисел в двоичной ПСС и трудность их восприятия человеком (см. табл. 2.1) приводит к необходимости перевода исходных данных (чисел) из десятичной системы счисления в двоичную, а результатов — из двоичной в десятичную. Эти переводы осуществляются в ЭВМ автоматически по определенным программам.

В соответствии с формулой (2.1) числа в разных ПСС можно представить следующим образом:

$$\begin{aligned} X_{(p_1)} &= a_n p_1^n + a_{n-1} p_1^{n-1} + \dots + a_0 p_1^0 + a_{-1} p_1^{-1} + \dots + \\ &+ a_{-m} p_1^{-m} = b_k p_2^k + b_{k-1} p_2^{k-1} + \dots + b_0 p_2^0 + b_{-1} p_2^{-1} + \\ &+ \dots + b_{-s} p_2^{-s} X_{(p_2)}. \end{aligned} \quad (2.2)$$

Следовательно, в общем виде задачу перевода числа из ПСС с основанием p_1 в ПСС с основанием p_2 , формулируемую в виде:

$$X_{(p_1)} = X_{(p_2)}; \quad X_{(p_2)} = ? \quad (2.3)$$

можно представить как задачу определения коэффициентов b_j нового ряда, изображающего число в ПСС с основанием p_2 .

Решение (2.3) непосредственно в виде (2.2) осуществляют при «ручном» способе перевода чисел из одной ПСС в другую. Для «машинной» реализации перевода применяют следующие методы:

Метод деления. Для перевода *целого числа* из десятичной системы счисления в любую другую ПСС необходимо разделить десятичное число на основание новой системы счисления, затем полученное частное снова разделить на основание новой системы счисления и так до тех пор, пока в частном не останется число меньше основания новой системы счисления (p_2). Число в новой системе счисления запишется в виде остатков от деления, начиная с последнего частного, представляющего собой старшую цифру числа. Поясним это на примерах.

Задача 2.1. $1967_{(10)} = X_{(8)}; X_{(8)} - ?$

Решение:

$$\begin{array}{r}
 1967 \overline{) 8} \\
 \underline{-16} 245 8 \\
 36 24 30 8 \\
 32 5 24 3 \\
 47 6 \\
 40 \\
 7
 \end{array}
 \quad
 \begin{array}{l}
 \text{Ответ:} \\
 1967_{(10)} = 3657_{(8)}.
 \end{array}$$

Задача 2.2. $375_{(10)} = X_{(2)}; X_{(2)} - ?$

Решение:

$$\begin{array}{r}
 375 \overline{) 2} \\
 \underline{-374} 1 187 2 \\
 1 186 93 2 \\
 1 92 46 2 \\
 1 46 23 2 \\
 0 22 5 2 \\
 1 4 2 2 \\
 1 2 2 \\
 0
 \end{array}$$

Ответ: $375_{(10)} = 101110111_{(2)}$.

Задача 2.3. $375_{(10)} = X_{(8)}; X_{(8)} - ?$

$$\begin{array}{r}
 375 \overline{) 8} \\
 \underline{-368} 46 8 \\
 7 40 5 \\
 6
 \end{array}$$

Ответ: $375_{(10)} = 567_{(8)}$.

Метод умножения. Для перевода *правильной десятичной дроби* в другую систему счисления необходимо дробную часть десятичного числа последовательно умножать на основание новой системы счисления, представленное в исходной ПСС, до тех пор, пока в дробной части не останутся нули или не будет достигнута заданная точность перевода. В результате выполнения каждой опе-

рации умножения формируется одна цифра нового числа (начиная со старшей), равная целой части очередного произведения. Рассмотрим на примерах.

Задача 2.4. $0,125_{(10)} = X_{(2)}$; $X_{(2)} - ?$

Решение:

$$\begin{array}{r}
 \times 0,125 \\
 \hline
 2 \\
 \times 0,250 \\
 \hline
 2 \\
 \times 0,500 \\
 \hline
 2 \\
 \hline
 1,000
 \end{array}$$

Ответ: $0,125_{(10)} = 0,001$

Задача 2.5. $0,125_{(10)} = X_{(16)}$; $X_{(16)} - ?$

Решение: $\times \begin{array}{r} 0,125 \\ 16 \\ \hline 2,000 \end{array}$

Ответ: $0,125_{(10)} = 0,2_{(16)}$.

Задача 2.6. $0,644_{(10)} = X_{(8)}$; $X_{(8)} - ?$

Решение: $\begin{array}{r} 0,644 \\ \times 8 \\ \hline 5,152 \\ \times 8 \\ \hline 1,216 \\ \vdots \end{array}$

Погрешность
преобразования

Ответ: $0,644_{(10)} \approx 0,51_{(8)}$.

Неправильные дроби десятичной системы счисления в любую другую переводятся в два приема: целая часть переводится по одному правилу, дробная — по другому. Затем целую и дробную части числа записывают вместе, отделяя запятой.

Задача 2.7. $191,644_{(10)} = X_{(8)}$; $X_{(8)} - ?$

Решение: $191_{(10)} = 277_{(8)}$; $0,644_{(10)} \approx 0,51_{(8)}$.

Ответ: $191,644_{(10)} \approx 277,51_{(8)}$.

Перевод чисел в десятичную ПСС из любой другой ПСС удобнее всего производить, представляя эти числа в развернутой форме (2.1):

$$1101001_{(2)} = 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 64 + 32 + 0 + 8 + 0 + 0 + 1 = 105_{(10)}.$$

Правила перевода восьмеричных и шестнадцатеричных чисел в двоичные и наоборот исключительно просты, поскольку основания восьмеричной и шестнадцатеричной систем счисления есть целые степени числа ($8 = 2^3$; $16 = 2^4$).

Для перевода восьмеричного (шестнадцатеричного) числа в двоичную форму достаточно заменить каждую цифру этого числа трехразрядным (четырёхразрядным) двоичным числом:

$$\begin{array}{l} 2516,1_{(8)} = 010101001110,001_{(2)}; \\ \dots 7B3,E_{(16)} = 011110110011,1110_{(2)}. \end{array}$$

При переводе из двоичной в восьмеричную (шестнадцатеричную) систему поступают следующим образом: двигаясь от запятой влево и вправо, разбивают двоичное число на группы по три (четыре) разряда, дополняя при необходимости нулями крайние левую и правую группы. Затем каждую группу из трех (четырех) разрядов заменяют соответствующей восьмеричной (шестнадцатеричной) цифрой:

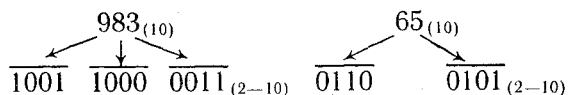
$$\begin{array}{l} 011001111,110100_{(2)} = 317,64_{(8)}; \\ 00110001,10111000_{(2)} = 31,B8_{(16)}. \end{array}$$

Заметим, что восьмеричная и шестнадцатеричная ПСС непосредственно не используются в ЭВМ для реализации операций, а служат лишь для облегчения чтения и записи человеком-оператором «машинных» кодов и широко используются в технике программирования. Очевидно, что самой удобной для человека является десятичная ПСС, а для машины — двоичная. Рассмотренные методы

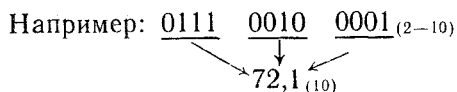
перевода чисел из одной ПСС в другую (делением и умножением) при реализации требуют существенных временных или аппаратных затрат. Для преодоления этого пользуются простым и оригинальным приемом: предварительно в двоичную ПСС переводят не все число, а только его цифры — в результате получается некоторая смешанная *двоично-десятичная система*.

Двоично-десятичное число образуется следующим образом: каждая цифра десятичного числа заменяется четырехразрядным двоичным числом — тетрадой.

Например:



При обратном преобразовании необходимо каждую тетраду заменить эквивалентной ей десятичной цифрой.



2.2. Структура и формат данных

Структуру данных определяют перечень и взаимосвязь всех сведений, необходимых для представления данных в разрядной сетке машины (величина разрядной сетки определяется количеством разрядов сумматора арифметико-логического устройства или ячейки памяти ЭВМ). В общем случае структура данных задается формой представления данных, для чисел — естественной и нормальной формами.

При естественной форме число записывается в естественном натуральном виде с выделением в общем случае следующих компонент числа: знака, запятой и цифр числа, например 238 — целое отрицательное число, 0,00125 — правильная дробь и т. д. Следовательно, для представления таких чисел в ЭВМ необходимо в слове данных выделить *поле знака*, *поле запятой* и *поле цифр*. Для сокращения длины разрядной сетки и упрощения обработки данных в конкретных ЭВМ положение запятой фиксируется схемотехнически, т. е. аппаратными

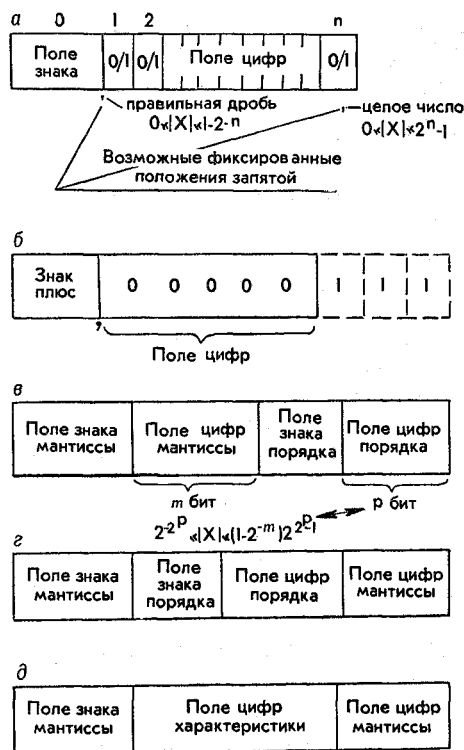


Рис. 2.1

средствами — представление чисел с фиксированной запятой. При этом в слове данных сохраняются лишь два структурных компонента: поле знака и поле цифр (рис. 2.1, а). Очевидно, что ограничение длины разрядной сетки (поля цифр) приводит к ограничению диапазона изменения чисел и потере точности их представления. Например, если под поле цифр выделено пять разрядов, то число 0,000001111 будет восприниматься ЭВМ как 0,00000 (рис. 2.1, б).

Для расширения диапазона представления чисел и уменьшения погрешности их задания используется нормальная форма записи.

При нормальной форме записи одного числа может принимать разный вид в зависимости от ограничений, накладываемых на ее форму. Например, число 2429 может быть записано так: $2429 = 2,429 \cdot 10^3 =$

Таблица 2.2

Тип ЭВМ	Формат данных	
	Фиксированная запятая	Плавающая запятая
1	2	3
ЕС ЭВМ	Слово (короткий формат):	
	<div> <div>знак</div> <div>поле цифр</div> </div> <div>0 1 31</div>	<div> <div>знак</div> <div>характеристика</div> <div>мантисса</div> </div> <div>0 1 8 31</div>
	Полуслово:	
	<div> <div>знак</div> <div>поле цифр</div> </div> <div>0 1 15</div>	<div> <div>знак</div> <div>характеристика</div> <div>мантисса (старшие разряды)</div> <div>мантисса (младшие разряды)</div> </div> <div>0 1 7 8 31</div>
Тройное слово (расширенный формат):		
<div> <div>знак</div> <div>характеристика</div> <div>мантисса (старшие разряды)</div> <div>мантисса (средние разряды)</div> <div>мантисса (младшие разряды)</div> </div> <div>0 1 7 8 31</div>		

$= 0,2429 \cdot 10^4 = 24290 \cdot 10^{-1}$ и т. д. В общем случае нормальная форма записи числа может быть представлена в виде $A = \pm tq^{\pm p}$, где t — мантисса числа; q — основание ПСС; p — порядок числа.

Порядок (с учетом знака) показывает, на сколько разрядов и в какую сторону сдвинута запятая при замене формы записи числа с естественной на нормальную. Поэтому такую форму записи называют *представление чисел с плавающей запятой*.

Для представления числа в нормальной форме в слове данных выделяют четыре структурных компонента (рис. 2.1, в, г): два поля знаков (мантиссы и порядка) и два поля цифр (поле q при выбранной ПСС избыточно).

Нетрудно убедиться, что в условиях ограничения разрядной сетки машины максимально возможную точность представления чисел (из рассмотренных) имеет нормальная форма записи числа, при которой старшая цифра мантиссы является значащей. Например: $0,125 \cdot 10^3_{(10)}$; $0,11011 \cdot 10^{101}_{(2)}$. Такие числа называют *нормализованными*.

Для упрощения операций над порядками (см. § 2.4 — 2.6) применяют *представление чисел с плавающей запятой со смещенным порядком*:

$$P_{\text{см}} = P + N,$$

где N — целое положительное число — смещение, $N = -\max(-P)$. Обычно $N = 2^k$, где k — число двоичных разрядов в поле цифр несмещенного порядка. В этом случае поле знаков порядков оказывается избыточным, так как код $P_{\text{см}}$ всегда положительный (рис. 2.1, д). Для устранения неоднозначности смещенные порядки называют *характеристиками*.

Формат данных определяет разрядность размещаемых в разрядной сетке машины структурных компонентов данных и выбирается с учетом принятых ПСС, способа кодирования структурных компонентов и диапазона изменения чисел.

Основной структурной единицей информации в машинах является бит (от англ. binary digit — двоичная цифра) — такое количество информации, которое доставляется получателю при приеме одного из двух возможных равновероятных сообщений (кодируемых, например, 0 и 1). В ЭВМ, использующей двоичную ПСС, бит представляет такое количество информации, которое может быть

1

2

3

СМ ЭВМ,
«Электроника-60М», ДВК

Слово:

знак	поле цифр
15	14 0

Двойное слово:

Знак	старшие разряды
	младшие разряды
15	14 0

Слово одинарной точности:

знак	характеристика	мантисса (старшие разряды)
	мантисса (младшие разряды)	
15	14 7 6	0

Слово двойной точности

знак	характеристика	мантисса (старшие разряды)
	мантисса (средние разряды)	
	мантисса (средние разряды)	
	мантисса (младшие разряды)	
15	14 7 6	0

Целое слово:

знак	поле цифр
15	14 0

Короткое вещественное слово:

знак	характеристика	мантисса (старшие разряды)
	мантисса (младшие разряды)	
15	14 7 6	0

ЕС 1841

1	2	3																					
Короткое целое:		Длинное вещественное слово:																					
<table><tr><td>знак</td><td>старшие разряды</td></tr><tr><td></td><td>младшие разряды</td></tr><tr><td>15</td><td>14 0</td></tr></table>		знак	старшие разряды		младшие разряды	15	14 0	<table><tr><td>знак</td><td>характеристика</td><td>мантисса (старшие разряды)</td></tr><tr><td></td><td></td><td>мантисса (средние разряды)</td></tr><tr><td></td><td></td><td>мантисса (средние разряды)</td></tr><tr><td></td><td></td><td>мантисса (младшие разряды)</td></tr><tr><td>15</td><td>14 4 3</td><td>0</td></tr></table>	знак	характеристика	мантисса (старшие разряды)			мантисса (средние разряды)			мантисса (средние разряды)			мантисса (младшие разряды)	15	14 4 3	0
знак	старшие разряды																						
	младшие разряды																						
15	14 0																						
знак	характеристика	мантисса (старшие разряды)																					
		мантисса (средние разряды)																					
		мантисса (средние разряды)																					
		мантисса (младшие разряды)																					
15	14 4 3	0																					
Длинное целое:		Временное вещественное слово:																					
<table><tr><td>знак</td><td>старшие разряды</td></tr><tr><td></td><td>средние разряды</td></tr><tr><td></td><td>младшие разряды</td></tr><tr><td>15</td><td>14 0</td></tr></table>		знак	старшие разряды		средние разряды		младшие разряды	15	14 0	<table><tr><td>знак</td><td>характеристика</td></tr><tr><td></td><td>мантисса (старшие разряды)</td></tr><tr><td></td><td>мантисса (средние разряды)</td></tr><tr><td></td><td>мантисса (средние разряды)</td></tr><tr><td></td><td>мантисса (младшие разряды)</td></tr><tr><td>15</td><td>14 0</td></tr></table>	знак	характеристика		мантисса (старшие разряды)		мантисса (средние разряды)		мантисса (средние разряды)		мантисса (младшие разряды)	15	14 0	
знак	старшие разряды																						
	средние разряды																						
	младшие разряды																						
15	14 0																						
знак	характеристика																						
	мантисса (старшие разряды)																						
	мантисса (средние разряды)																						
	мантисса (средние разряды)																						
	мантисса (младшие разряды)																						
15	14 0																						

записано в одном разряде машинной сетки. Поэтому бит часто используют в качестве единицы измерения длины машинного слова, емкости памяти и пр. Эволюция вычислительной и информационной техники вызвала появление 8-битовой единицы измерения, которая получила название байт. Большинство современных ЭВМ оперируют числами из 8, 16, 24 или 32 бит, или 1, 2, 3 или 4 байт. Для расширения диапазона представляемых чисел последние могут занимать два — четыре соседних слова (табл. 2.2). Кроме того, в некоторых типах ЭВМ используются специальные приемы расширения диапазона представляемых чисел. Например, в системе малых ЭВМ (СМ ЭВМ) старший разряд мантиссы не представляется в явном виде в поле цифр мантиссы, так как его содержимое всегда равно единице (числа нормализованные), а это может быть учтено схемотехнически при выполнении операций над числами. Так увеличивается на один разряд длина поля цифр мантиссы.

2.3. Кодирование чисел

Необходимость кодирования чисел определяется тем, что в ЭВМ нельзя либо нерационально вводить числа в том виде, в котором они изображаются человеком на бумаге.

Простейшим машинным кодом является прямой код $[X]_{пр}$, получаемый при кодировании в числе X только знаковой информации, причем знак «+» кодируется нулем, а знак «-» — единицей.

Например: $X_1 = -10011_{(2)}$, $[X_1]_{пр} = 110011$;

$$\begin{array}{l} \uparrow \\ \text{код знака «-»} \\ X_2 = -0,11011_{(2)}, [X_2]_{пр} = 1,11011; \\ X_3 = +0,1101_{(2)}, [X_3]_{пр} = 0,1101. \\ \uparrow \\ \text{код знака «+»} \end{array}$$

Подчеркнем, если под поле цифр разрядов выделено больше, чем это необходимо для представления числа X , то разряды (цифры) числа X заносятся в разрядную сетку ЭВМ в соответствии со своими весами. Код знака числа практически во всех ЭВМ заносится в старший разряд разрядной сетки (рис. 2.1, табл. 2.2).

Следовательно, при использовании, например, 8-раз-

рядной сетки числа $[X_1]_{\text{пр}} = 10010011$; $[X_2]_{\text{пр}} = 1,1101100$; $[X_3]_{\text{пр}} = 0,1101000$.

Прямой код числа широко используется в ЭВМ при хранении чисел в памяти и выполнении операций умножения. Алгебраическое (с учетом знаков) сложение чисел в прямом коде затруднено, так как требуется выполнить следующие четыре действия: 1) сравнить знаки слагаемых; 2) сравнить слагаемые по модулю при неравенстве их знаков; 3) выполнить сложение модулей слагаемых (при равенстве знаков) или вычитание из большего по модулю меньшего слагаемого (при неравенстве знаков); 4) присвоить алгебраической сумме знак большего по модулю слагаемого.

Для упрощения алгоритма алгебраического сложения используют несколько усложненное кодирование отрицательных чисел: обратный и дополнительный коды.

Обратный код $[X]_{\text{обр}}$ отрицательного числа формируется по следующему правилу: в знаковом разряде представляется единица, а во всех остальных разрядах цифры заменяются на взаимно обратные.

Для рассматриваемых примеров при 6-разрядной сетке: $[X_1]_{\text{обр}} = 101100$; $[X_2]_{\text{обр}} = 1,00100$, а при 8-разрядной сетке: $[X_1]_{\text{обр}} = 11101100$; $[X_2]_{\text{обр}} = 1,1100100$.

Дополнительный код $[X]_{\text{доп}}$ отрицательного числа образуется из обратного кода путем увеличения $[X]_{\text{обр}}$ на единицу младшего разряда. При 6- и 8-разрядных сетках дополнительные коды чисел X_1 и X_2 имеют соответственно вид:

$$[X_1]_{\text{доп}} = 101101; [X_2]_{\text{доп}} = 1,00101$$

и

$$[X_1]_{\text{доп}} = 11101101; [X_2]_{\text{доп}} = 1,1100101.$$

Заметим, что прямой, обратный и дополнительный коды положительных чисел совпадают. Поэтому многие не делают различий между ними, полагая, что положительное число имеет единственное изображение в ЭВМ — прямой код.

ЭВМ способны обрабатывать как числовую, так и текстовую информацию. Текстовая информация (русские и латинские буквы, знаки препинания и пр.), как и числа, кодируется последовательностью двоичных цифр. При использовании для этих целей 8-разрядных кодов (байтов) можно закодировать 256 различных символов (табл. 2.3). Такая система кодирования носит название КОИ-7 и принята в большинстве микроЭВМ.

Таблица 2.3

Код символа в 16-ричной системе счисления						
Младшая цифра	Старшая цифра					
	2	3	4	5	6	7
0	Пробел	0	@	P	Ю	П
1	!	1	A	Q	А	Я
2	»	2	B	R	Б	Р
3	#	3	C	S	Ц	С
4	№	4	D	T	Д	Т
5	%	5	E	U	Е	У
6	&	6	F	V	Ф	Ж
7	'	7	G	W	Г	В
8	(8	H	X	Х	Ь
9)	9	I	Y	И	Ы
A	*	:	J	Z	Й	З
B	+	;	K	[К	Ш
C	,	<	L	\	Л	Э
D	-	=	M]	М	Ш
E	.	>	N	^	Н	Ч
F	/	?	O	—	О	Забой

2.4. Сложение двоичных чисел

Арифметические действия над двоичными числами производятся по тем же правилам, что и над десятичными. Необходимо только учитывать, что сложение двух единиц дает нуль в данном разряде и единицу переноса в следующий.

Задача 2.8. Найти сумму двух чисел $X_1 = 1101_{(2)}$ и $X_2 = 101_{(2)}$. В дальнейшем, для упрощения записи, указанную задачу будем формулировать в следующем виде: $X_1 = 1101$; $X_2 = 101$; $X_1 + X_2 = ?$

Решение:

$$\begin{array}{r} + 1101_{(2)} \\ + 101_{(2)} \\ \hline 10010_{(2)} \end{array}$$

Ответ: $X_1 + X_2 = 10010_{(2)}$.

Сложение чисел с фиксированной запятой с различными знаками благодаря использованию обратного или дополнительного кода для отрицательных чисел сводится в ЭВМ к арифметическому сложению кодов чисел. Знаковые разряды участвуют в операции сложения наравне с цифровыми. При этом, если выполняется операция сложения в обратных кодах, единица переноса из знакового разряда суммы прибавляется к ее младшему разряду

(т. е. выполняется циклический перенос). Если же операция сложения выполняется над числами, представленными в дополнительном коде, то единица переноса из знакового разряда суммы отбрасывается. Переход от обратного и дополнительного кодов к прямому выполняется аналогично переходу от прямого кода к обратному и дополнительному соответственно.

Рассмотрим примеры сложения чисел в обратном и дополнительном кодах.

Задача 2.9. $[X_1]_{\text{обр}} = 0,001001$; $[X_2]_{\text{обр}} = 1,001110$; $[X_1 + X_2]_{\text{обр}} = ?$;
 $X_1 + X_2 = ?$

Решение:

$$\begin{array}{r} + 0,001001 \\ + 1,001110 \\ \hline 1,010111 \end{array}$$

Ответ: $[X_1 + X_2]_{\text{обр}} = 1,010111$; $X_1 + X_2 = -0,101000$.

Комментарий. Результат сложения чисел, представленных в обратном коде, также получился в обратном коде (сумма отрицательна). Для перевода числа из обратного кода в прямой значения цифровых разрядов меняются на противоположные.

Задача 2.10. $[X_1]_{\text{обр}} = 0,110001$; $[X_2]_{\text{обр}} = 1,110110$; $X_1 + X_2 = ?$

Решение:

$$\begin{array}{r} + 0,110001 \\ + 1,110110 \\ \hline 10,100111 \\ + \quad \quad \quad 1 \\ \hline 0,101000 \end{array}$$

Ответ: $X_1 + X_2 = +0,101000$.

Комментарий. Для формирования правильного результата необходимо выполнить циклический перенос единицы. Сумма положительна, следовательно, результат соответствует прямому коду суммы.

Задача 2.11. $[X_1]_{\text{обр}} = 1,001110$; $[X_2]_{\text{обр}} = 1,110110$; $X_1 + X_2 = ?$

Решение:

$$\begin{array}{r} + 1,001110 \\ + 1,110110 \\ \hline 11,000100 \\ + \quad \quad \quad 1 \\ \hline 1,000101 \end{array}$$

Ответ: $X_1 + X_2 = -0,111010$.

Комментарий. Для определения суммы осуществлены циклический перенос единицы и преобразование кодов из обратного в прямой.

Задача 2.12. $[X_1]_{\text{доп}} = 0,001001$; $[X_2]_{\text{доп}} = 1,001111$; $X_1 + X_2 = ?$

Решение:

$$\begin{array}{r} + 0,001001 \\ + 1,001111 \\ \hline 1,011000 \end{array}$$

Ответ: $X_1 + X_2 = -0,101000$.

Комментарий. Для определения результата осуществлено преобразование кодов из дополнительного в прямой:
 $[1,011000]_{\text{доп}} = [1,101000]_{\text{пр}} = -0,101000$.

Задача 2.13. $[X_1]_{\text{доп}} = 0,110001$; $[X_2]_{\text{доп}} = 1,110111$; $X_1 + X_2 = ?$

$$\begin{array}{r} \text{Решение:} \\ 0,110001 \\ + 1,110111 \\ \hline 10,101000 \end{array}$$

← отбрасывается

Ответ: $X_1 + X_2 = 0,101000$.

Задача 2.14. $[X_1]_{\text{доп}} = 1,001111$; $[X_2]_{\text{доп}} = 1,110111$; $X_1 + X_2 = ?$

$$\begin{array}{r} \text{Решение:} \\ 1,001111 \\ + 1,110111 \\ \hline 11,000110 \end{array}$$

← отбрасывается

Ответ: $X_1 + X_2 = -0,111010$.

Как видно из примеров, сложение с использованием обратных кодов, как и в случае дополнительных кодов, выполняется по обычным правилам сложения двоичных чисел (при этом знаковый разряд рассматривается наравне с цифровыми), с той лишь разницей, что в случае обратных кодов в процессе получения суммы реализуется (при необходимости) циклический перенос, тогда как при дополнительных кодах единица переноса из знакового разряда не учитывается (отбрасывается). Следовательно, в общем виде реализация операции сложения в дополнительных кодах проще по сравнению с реализацией сложения в обратных кодах. Однако достигается это за счет усложнения операции кодирования чисел.

Следует также иметь в виду, что при сложении двух одинаковых по абсолютной величине чисел с разными знаками в случае использования обратных кодов получается отрицательный ноль ($1,11\dots11$ либо $1,11\dots1$). В ЭВМ отрицательный ноль автоматически преобразуется в положительный (т. е. к виду $0,00\dots0$ при сложении правильных дробей либо $0,00\dots0$ при сложении целых чисел соответственно). При сложении двух чисел с одинаковыми знаками возможно переполнение разрядной сетки сумматора (устройства, реализующего операцию сложения). Это, очевидно, приводит к существенному искажению результата. Покажем это на примерах.

Задача 2.15. $[X_1]_{\text{пр}} = 0,11010$; $[X_2]_{\text{пр}} = 0,01010$; $X_1 + X_2 = ?$

$$\begin{array}{r} \text{Решение:} \\ 0,11010 \\ + 0,01010 \\ \hline 1,00100 \end{array}$$

Ответ: $X_1 + X_2 = -0,00100$.

Комментарий. Ответ ошибочный (при сложении двух положительных чисел сумма получилась отрицательная). Результат не совпадает с истинным ни по знаку, ни по абсолютной величине.

Задача 2.16. $[X_1]_{\text{доп}} = 1,0101$; $[X_2]_{\text{доп}} = 1,0011$; $X_1 + X_2 = ?$

Решение:

$$\begin{array}{r} 1,0101 \\ + 1,0011 \\ \hline 10,1000 \end{array}$$

← отбрасывается

Ответ: $X_1 + X_2 = +0,1000$.

Комментарий. Ответ ошибочный. Правильный результат $X_1 + X_2 = -1,1000$.

Задача 2.17. $[X_1]_{\text{обр}} = 1,1001$; $[X_2]_{\text{обр}} = 1,0010$; $X_1 + X_2 = ?$

Решение:

$$\begin{array}{r} 1,1001 \\ + 1,0010 \\ \hline 10,1011 \\ \boxed{+} \rightarrow 1 \\ \hline 0,1100 \end{array}$$

Ответ: $X_1 + X_2 = +0,1100$.

Комментарий. Ответ ошибочный. Должно быть $X_1 + X_2 = -1,0011$.

Знак суммы (содержимое знакового разряда) может измениться по отношению к знаку слагаемых лишь в том случае, если при выполнении операции сложения имел место перенос либо только из знакового разряда суммы (при сложении отрицательных чисел), либо только в знаковый разряд (при сложении положительных чисел). При наличии переносов из знакового разряда и в знаковый разряд знак суммы совпадает со знаком слагаемых. Поэтому в ЭВМ для обнаружения факта переполнения разрядной сетки сумматора анализируются не знаки, а лишь факт наличия (отсутствия) переноса из знакового разряда и в знаковый разряд.

Следует иметь в виду, что при выполнении операции сложения в дополнительном коде возможна ситуация (единственная), когда указанное правило не различает переполнение. Это происходит тогда, когда сумма модулей двух отрицательных чисел равна удвоенному весу единицы старшего разряда числа (для правильной дроби — единице).

Задача 2.18. $[X_1]_{\text{доп}} = 1,0111$; $[X_2]_{\text{доп}} = 1,1001$; $[X_1 + X_2]_{\text{доп}} = ?$

Решение:

$$\begin{array}{r} 1,0111 \\ + 1,1001 \\ \hline 11,0000 \end{array}$$

← отбрасывается

Ответ: $[X_1 + X_2]_{\text{доп}} = 1,0000$.

Комментарий. Ответ ошибочный. В рассмотренном примере знаки суммы и слагаемых совпадают, при выполнении операции сложения осуществлялись переносы в знаковый разряд и из знакового разряда. Однако переполнение все же возникло. Для его обнаружения в ЭВМ необходимо предусмотреть анализ на ноль содержимого цифровых разрядов суммы.

В некоторых ЭВМ для обнаружения переполнения применяют так называемые *модифицированные коды* (прямой, обратный и дополнительный), которые отличаются от основных (немодифицированных) лишь тем, что под поле знаков выделены два разряда. Например, число $X_1 = +0,11011$ в прямом модифицированном коде имеет вид $[x_1]_{\text{пр}}^{\text{м}} = 00,11011$; число $X_2 = -0,10110$ в обратном модифицированном коде записывается как $[X]_{\text{обр}}^{\text{м}} = 11,01001$ и пр.

Сложение чисел в модифицированных кодах выполняется по тем же правилам, что и в основных кодах. Признаком переполнения здесь является различие содержимых знаковых разрядов: комбинация 01 фиксирует переполнение при сложении двух положительных чисел (положительное переполнение), а 10 — двух отрицательных (отрицательное переполнение). Заметим, что та же особая ситуация, возникающая при сложении отрицательных чисел в дополнительном коде, не различается и в модифицированном дополнительном коде.

Если числа X_1 и X_2 представлены в форме с плавающей запятой, т. е. $X_1 = m_1 \cdot 2^{p_1}$, $X_2 = m_2 \cdot 2^{p_2}$, то для их сложения необходимо X_1 и X_2 привести к общему порядку $P_{\text{общ}}$. Если, например, $P_{\text{общ}} = P_1$, то приведение к общему порядку отразится лишь на X_2 :

$$X_2 = m_2 2^{p_2} = m_2' 2^{p_1}.$$

Далее можно общий множитель вынести за скобки и провести сложение мантисс:

$$X_1 + X_2 = 2^{p_1} (m_1 + m_2').$$

Очевидно, при конечной длине поля цифр мантиссы (табл. 2.2) $P_{\text{общ}}$ необходимо выбирать из условия $P_{\text{общ}} = \max\{P_1, P_2\}$, так как в противном случае произойдет переполнение разрядной сетки мантиссы преобразуемого числа (в памяти машины числа с плавающей запятой хранятся в нормализованном виде).

В ЭВМ определен следующий порядок выполнения операции сложения чисел, представленных в форме с плавающей запятой.

1. Вычитание порядков складываемых чисел. Если разность порядков равна нулю (порядки равны), то перейти к пункту 3. Если порядки не равны, то осуществить выравнивание порядков (пункт 2).

2. Увеличение меньшего из порядков до большего.

Мантисса числа с меньшим порядком сдвигается вправо на число разрядов, равное разности порядков.

3. Сложение мантисс. Если не произошло переполнения разрядной сетки мантиссы и сумма получена в нормализованном виде, то вычисления закончить. Сумма мантисс является мантиссой суммы чисел, а порядок суммы чисел равен порядку большего числа (или общему порядку). В противном случае перейти к пункту 4.

4. Нормализация полученной суммы вправо (при переполнении разрядной сетки мантиссы) или влево (при наличии в мантиссе нулей). При нормализации вправо мантиссу суммы сдвинуть вправо на один разряд, а порядок суммы увеличить на единицу. При нормализации влево мантиссу суммы сдвинуть влево до первой значащей цифры, а порядок суммы уменьшить на такое же количество единиц.

Задача 2.19. $X_1 = 0,1011001 \cdot 2^{011}$; $X_2 = 0,1001101 \cdot 2^{011}$; $X_1 + X_2 = ?$

Решение. Выполним пункт 1:

$P_1 = 0,011$; $P_2 = 0,011$; $P_{\text{общ}} = ?$

код знака

Заменим разность положительных чисел $P_1 - P_2$ суммой чисел с разным знаком $P_1 + (-P_2)$:

$[P_1]_{\text{обр}} = 0,011$

+ +

$[P_2]_{\text{обр}} = 1,100$

$[P_1 + P_2]_{\text{обр}} = 1,111$ — отрицательный ноль.

Так как $P_1 - P_2 = 0$, то $P_1 = P_2 = 0,011$. Поэтому переходим к реализации пункта 3.

$m_1 = 0,1011001$; $m_2 = 0,1001101$; $m_1 + m_2 = ?$

+ 0,1011001

+ 0,1001101

1,0100110

При сложении m_1 и m_2 произошло переполнение разрядной сетки мантиссы (денормализация числа влево), признаком которого явились наличие переноса в знаковый разряд и отсутствие переноса из знакового разряда. Следовательно, выполняем пункт 4.

Для нормализации суммы сдвинем мантиссу на 1 разряд вправо, мантисса уменьшится на порядок (в два раза). Одновременно для сохранения величины суммы увеличим на единицу код порядка. После нормализации сумма имеет вид $X_1 + X_2 = 0,1010011 \cdot 2^{100}$.

Задача 2.20. $X_1 = 0,1011001 \cdot 2^{011}$; $X_2 = -0,1001101 \cdot 2^{011}$; $X_1 + X_2 = ?$

Решение. Пункт 1 аналогичен рассмотренному выше. Пункт 2 также не выполняется.

3. $m_1 = 0,1011001$; $m_2 = -0,1001101$; $m_1 + m_2 = ?$

+ 0,1011001

+ 1,0110010

10,0001011

+ 1

0,0001100

$m_1 + m_2 = 0,0001100$

Произошла денормализация числа вправо.

4. Для нормализации числа полученная в пункте 3 сумма сдвигается на три разряда влево (увеличивается в 2^3 раза), а код порядка, сформированный в пункте 1, уменьшается на три, т. е. $P_{\text{обн}} = 011 - 011 = 0$. Следовательно, $X_1 + X_2 = 0,1100000 \cdot 2^0$.

Задача 2.21. $X_1 = 0,1101101 \cdot 2^{011}$; $X_2 = 0,1110001 \cdot 2^{001}$; $X_1 + X_2 = ?$

Решение: 1. $P_1 - P_2 = 10$.

2. $P'_2 = 011$; $m'_2 = 0,001110001$

отбрасывается

При денормализации числа вправо происходит потеря точности его представления из-за ограничения разрядной сетки.

3. $m_1 + m'_2 = ?$

$$\begin{array}{r} 0,1101101 \\ + 0,0011100 \\ \hline 1,0001001 \end{array}$$

Произошла денормализация числа влево.

4. После нормализации $X_1 + X_2 = 0,1000100 \cdot 2^{100}$.

2.5. Умножение двоичных чисел

Применительно к двоичной ПСС наиболее известны следующие основные способы выполнения операций умножения:

умножение начиная с младших разрядов множителя:

а) «ручной» метод:

$$\begin{array}{r} \times 1001 \text{ — множимое,} \\ 0101 \text{ — множитель,} \\ \hline 1001 \\ + 0000 \text{ — частичные} \\ 1001 \text{ — произведения,} \\ \hline 0000 \\ \hline 0101101 \text{ — произведение;} \end{array}$$

б) «машинный» метод:

$$\begin{array}{r} \times 1001 \text{ — множимое,} \\ 0101 \text{ — множитель,} \\ \hline 1001 \text{ — частичные} \\ + 01001 \text{ — суммы,} \\ \hline + 101101 \\ \hline 0101101 \text{ — произведение;} \end{array}$$

умножение начиная со старших разрядов множителя:

а) «ручной» метод:

$$\begin{array}{r} 1001 \text{ — множимое,} \\ \times 0101 \text{ — множитель,} \\ \hline 0000 \\ 1001 \text{ — частичные} \\ 0000 \text{ — произведения,} \\ 1001 \\ \hline 0101101 \text{ — произведение;} \end{array}$$

б) «машинный» метод:

1001	—	множимое,
0101	—	множитель,
+ 0000	\swarrow \rightarrow \swarrow \rightarrow	частичные суммы,
+ 01001		
+ 010010		
+ 0101101		
		— произведение.

При «ручном» методе в обоих случаях умножение сводится к последовательному поразрядному умножению множимого на цифры множителя и накоплению (суммированию) получаемых частичных произведений. При этом операциями сложения могут управлять разряды множителя: если в i -м разряде множителя находится единица, то к сумме частичных произведений добавляется множимое с соответствующим сдвигом на $i - 1$ разряд (вправо или влево в зависимости от принятого способа выполнения операции умножения); если в i -м разряде множителя нуль, то множимое не прибавляется.

При «машинном» методе произведение формируется в виде возрастающего (по модулю) значения частичной суммы, равной после умножения на i -й разряд множителя сумме первых i частичных произведений.

Рассмотренные примеры показывают, что для получения произведений, помимо сложения, необходимо выполнять операции сдвига чисел (множимого либо частичной суммы). Очевидно, что произведение двух n -разрядных чисел есть число $2n$ -разрядное. Поэтому в случае ограничения поля цифр произведения n разрядами при умножении целых чисел в качестве результата берутся младшие n разрядов (в старших n разрядах должны быть нули, так как в противном случае вырабатывается признак переполнения), а при умножении правильных дробей в качестве результата берутся старшие n разрядов (младшие n разрядов отбрасываются — происходит усечение числа — либо используются для округления кода старших n разрядов).

Знак произведения формируется по известному правилу: $(+) \cdot (+) = (+)$; $(+) \cdot (-) = (-)$; $(-) \cdot (+) = (-)$; $(-) \cdot (-) = (+)$. В ЭВМ, как было указано в § 2.3, знак «—» числа кодируется единицей, а «+» — нулем, но правило формирования знака сохраняется. Операция, которую реализует ЭВМ для определения знака произведения, называется *суммой по модулю два* и обозначается \oplus : $0 \oplus 0 = 0$; $0 \oplus 1 = 1$; $1 \oplus 0 = 1$; $1 \oplus 1 = 0$.

Задача 2.22. $[X_1]_{\text{пр}} = 0.1101$; $[X_2]_{\text{пр}} = 0.1101$; $[X_1 \cdot X_2]_{\text{пр}} = ?$

Решение: знак произведения: $0 \oplus 0 = 0$.

Перемножим модули чисел в прямом коде, начиная с младших разрядов множителя со сдвигом частичных сумм (суммы частичных произведений) вправо на один разряд:

Множимое:	1101
Множитель:	1101
Первое частичное произведение:	11010000
Первая частичная сумма С1:	11010000
Сдвиг С1 вправо:	01101000
Второе частичное произведение:	00000000
Вторая частичная сумма С2:	01101000
Сдвиг С2 вправо:	00110100
Третье частичное произведение:	11010000
Третья частичная сумма С3:	100000100*
Сдвиг С3 вправо:	10000010
Четвертое частичное произведение:	11010000
Четвертая частичная сумма С4:	101010010*
Сдвиг С4 вправо:	10101001

* В данном алгоритме возможно временное переполнение разрядной сетки.

Ответ: $[X_1 \cdot X_2]_{\text{пр}} = 0.10101001$.

Алгоритм умножения чисел, представленных в форме с плавающей запятой, определяется следующим соотношением:

$$(X_1 = m_1 \cdot 2^{p_1}) \cdot (X_2 = m_2 \cdot 2^{p_2}) = (m_1 + m_2) \cdot 2^{p_1 + p_2}.$$

При реализации операции умножения над числами с плавающей запятой выделяют следующие этапы:

- 1) определение знака произведения путем сложения по модулю два знаков мантисс операндов;
- 2) перемножение модулей мантисс по правилам умножения дробных чисел с фиксированной запятой;
- 3) определение порядка произведения путем алгебраического сложения порядков сомножителей (с использованием дополнительного, обратного или модифицированного кодов);
- 4) нормализация результата (так как сомножители нормализованы, то денормализация возможна только на 1 разряд и только вправо) и округление мантиссы в случае необходимости.

2.6. Деление двоичных чисел

Если умножение выполняется путем многократных сдвигов и сложений, то деление, будучи операцией обратной умножению,— путем многократных сдвигов и вычитаний.

При представлении чисел с фиксированной запятой деление возможно, если делимое по модулю меньше делителя, в противном случае произойдет переполнение разрядной сетки. Так же, как и при «ручном» делении, разряды частного при делении чисел на машине определяются (начиная со старшего) путем последовательного вычитания делителя из остатка, полученного от предыдущего вычитания. Однако здесь операция вычитания заменяется операцией сложения остатка с отрицательным делителем, представленным в обратном или дополнительном коде. Знак частного определяется сложением по модулю два кодов знаков делимого и делителя.

Рассмотрим сначала пример деления «ручным» способом.

Задача 2.23. $X_1 = +0,1001$; $X_2 = +0,1101$; $X_1 : X_2 = ?$

Решение:

$$\begin{array}{r}
 10010 \quad | \quad 1101 \\
 - 1101 \quad | \quad 0,1011 \\
 \hline
 10100 \\
 - 1101 \\
 \hline
 1110 \\
 - 1101 \\
 \hline
 0001
 \end{array}$$

Ответ: $X_1 : X_2 \approx 0,1011$.

Погрешность вычисления: $\frac{1}{1101}$.

Здесь после каждого вычитания делитель сдвигается вправо по отношению к делимому. Если остаток после вычитания получился положительный, в разряд частного записывается 1, если отрицательный — нуль. На практике обычно отрицательный остаток не записывается, просто делитель сдвигается дополнительно на один разряд вправо и вычитается из положительного остатка.

В машинах вместо сдвига делителя вправо осуществляется сдвиг остатка влево, что, по сути, ничего не изменяет.

При делении с восстановлением остатка отрицательный остаток восстанавливается суммированием с положительным делителем. Восстановленный остаток сдвигается влево на один разряд. Из сдвинутого остатка вновь вычитается делитель. По знаку полученного остатка определяется цифра очередного разряда частного. Процесс деления продолжается до получения заданного числа цифр частного, обеспечивающего необходимую точность результата.

Посмотрим, как решается предыдущий пример на машине.

Процесс деления начинается со сдвига делимого влево на один разряд, после чего к нему прибавляется делитель, представленный, например, в дополнительном модифицированном коде:

$[X_1]_{\text{доп}}^m$	00,1001	00,1101
		00,1011
Сдвиг влево	+ 01,0010	
$[-X_2]_{\text{доп}}$ (вычитание)	+ 11,0011	
Остаток положительный	00,0101	
Сдвиг влево	+ 00,1010	
Вычитание	+ 11,0011	
Остаток отрицательный	11,1101	
Восстановление	+ 00,1101	
	00,1010	
Сдвиг влево	+ 01,0100	
	+ 11,0011	
Остаток положительный	00,0111	
Сдвиг влево	+ 00,1110	
Вычитание	+ 11,0011	
Остаток положительный	00,0001	

Очевидно, что при делении с восстановлением остатка в самом неблагоприятном случае для формирования каждого разряда частного требуется выполнить две операции: вычитания (сложения в дополнительном или обратном коде) и сложения (восстановления остатка). То есть время выполнения операции деления может оказаться в два раза больше по сравнению с минимально возможным.

Для сокращения среднего времени выполнения операции деления реализуют деление без восстановления остатка, алгоритм которого следующий.

1. Определить знак частного суммированием по модулю два содержимых знаковых разрядов делимого и делителя.

2. Из делимого вычесть делитель. Если остаток отрицательный, перейти к пункту 3. В противном случае вычисление закончить (произошло переполнение).

3. Запомнить знак остатка.

4. Сдвинуть остаток на один разряд влево.

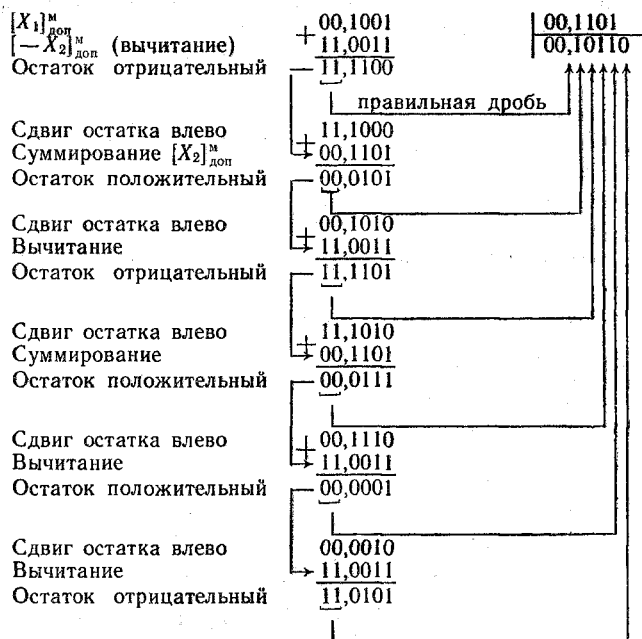
5. Присвоить делителю знак, обратный знаку остатка, запомненному в п. 2.

6. Сложить сдвинутый остаток и делитель (с учетом знака).

7. Присвоить цифре частного значение, противоположное коду знака остатка.

8. Повторять выполнение пунктов 3—7 до тех пор, пока не будет обеспечена требуемая точность вычисления частного.

Решение рассмотренного выше примера в данном случае осуществляется по следующей схеме:



Ответ: (с учетом округления) $X_1 : X_2 = 0,1011$.

При выполнении операции деления над числами с плавающей запятой мантисса частного определяется как результат деления мантиссы делимого на мантиссу делителя, а порядок частного в результате вычитания кода порядка делителя из кода порядка делимого, так как

$$(2^{p_1} M_1) : (2^{p_2} M_2) = 2^{p_1 - p_2} (M_1 : M_2).$$

Деление целых ненулевых n -разрядных (не считая

знаковых разрядов) чисел $A:B$, представленных в прямом (для простоты) коде, приводит к получению целого частного C и целого остатка 0 , которому присваивается знак делимого; знак частного вычисляется как сумма по модулю два операндов A и B .

Деление выполняется в следующей последовательности.

1. Делитель B сдвигается влево (нормализуется), так чтобы в старшем информационном разряде оказалась 1; подсчитывается количество сдвигов S ; частное от деления может быть не более $(S + 1)$ разрядов, не равных нулю.

2. Выполняется $(S + 1)$ цикл деления модулей $|A|$ на $|B'|$, где B' — нормализованное B , в результате находится $(S + 1)$ разряд частного, начиная со старшего из $(S + 1)$ младших.

3. Полученный в последнем цикле деления остаток R_{S+1} , если он положительный, сдвигается вправо на S разрядов; если же $R_{S+1} < 0$ (отрицательный), то остаток восстанавливается: к нему добавляется $|B'|$, т. е. $[R_{S+1}]_{\text{восст}} = R_{S+1} + |B'|$. После этого выполняется сдвиг вправо на S разрядов. В результате получается целый остаток от деления.

4. Частному и остатку присваиваются знаки.

Задача 2.24. $A = -27$; $B = 4$; $A:B = ?$

Решение. Осуществим перевод исходных операндов из десятичной ППС в двоичную:

$[A]_{\text{пр}} = 1.11011$; $[B]_{\text{пр}} = 0.00100$.

Выполним действия согласно приведенному описанию.

1. $B = 0.10000$; $S = 2$.

2. Выполняем деление $|A|$ на $|B'|$:

$$\begin{array}{r}
 |A| \quad + \quad 0.11011 \quad |0.10000| \\
 [-|B'|]_{\text{доп}} + \quad 1.10000 \quad 0.00110 \\
 R_1 > 0 \quad \underline{10.01011} \quad \uparrow \uparrow \uparrow \\
 \text{Сдвиг } R_1 \quad \underline{0.10110} \quad \uparrow \uparrow \uparrow \\
 [-|B'|]_{\text{доп}} + \quad 1.10000 \\
 R_2 > 0 \quad \underline{10.00110} \quad \uparrow \uparrow \uparrow \\
 \text{Сдвиг } R_2 \quad \underline{0.01100} \quad \uparrow \uparrow \uparrow \\
 [-|B'|]_{\text{доп}} + \quad 1.10000 \\
 R_3 < 0 \quad \underline{1.11100} \quad \uparrow \uparrow \uparrow
 \end{array}$$

3. Остаток $R_3 < 0$, поэтому прибавляем к нему $|B'|$:

$$\begin{array}{r}
 R_3 \quad + \quad 1.11100 \\
 |B'| \quad + \quad 0.10000 \\
 [R_3]_{\text{восст}} \quad \underline{10.01100}
 \end{array}$$

Сдвигаем $[R_3]_{\text{восст}}$ вправо на 2 разряда и получаем модуль остатка от деления:

$$|R| = 0.00011.$$

4. Присваиваем знаки частному и остатку, получаем результат:

$$[C]_{\text{пр}} = 1.00110; [R]_{\text{пр}} = 1.00011.$$

Ответ: $-27:4 = -6$ и остаток -3 .

Контрольные вопросы и задания

1. Чем является цифра 2 в десятичном числе 2364: а) наименьшей значащей десятичной цифрой, б) наибольшей значащей десятичной цифрой, в) наименьшим значащим битом, г) наибольшим значащим битом.

2. Что означает понятие «бит»?

3. Какое из двух чисел больше: $1111_{(2)}$ или $11_{(10)}$? Обоснуйте ответ.

4. Какую из указанных операций необходимо выполнять при преобразовании целой части числа из одной системы счисления в другую: а) сложение, б) вычитание, в) умножение, г) деление?

5. Преобразуйте в двоичные эквиваленты следующие десятичные числа:

а) 23; б) 105; в) 32; г) 15; д) 206; е) 128; ж) 63; з) 29; и) 12,125; к) 16,375; л) 2,5.

6. Приводимые ниже десятичные числа преобразуйте в двоичные, восьмеричные, шестнадцатеричные: а) 126; б) 4; в) 16; г) 63; д) 101; е) 12; ж) 1; з) 127; и) 9,25.

7. Определите диапазон представления положительных двоичных чисел в n -разрядной сетке машины, если запятая (точка) фиксирована после младшего разряда.

8. Представьте следующие числа в прямом, обратном, дополнительном машинных кодах: а) $-0,1101101$; б) $+0,1110110$; в) $-0,00001$; г) $-0,111011$.

9. Сложите числа с фиксированной запятой $[X]_{\text{пр}} = 0.01101$, $[Y]_{\text{пр}} = 0.0010$. Сложение произвести в модифицированном обратном коде.

10. Сложите числа с фиксированной запятой $[X]_{\text{пр}} = 0,1001$, $[Y]_{\text{пр}} = 1,1101$. Сложение чисел произвести в модифицированном дополнительном коде.

11. Сложите числа с фиксированной запятой $[X]_{\text{пр}} = 1,11001$, $[Y]_{\text{пр}} = 1,01111$. Сложение произвести в модифицированном дополнительном коде.

12. Вычитите из $[X]_{\text{пр}} = 0.0011$ $[Y]_{\text{пр}} = 1.01110$. Вычитание чисел произвести в модифицированном дополнительном коде.

13. Вычитите из $[X]_{\text{пр}} = 1,11001$ $[Y]_{\text{пр}} = 10101$. Вычитание произвести в модифицированном обратном коде.

14. Выполните умножение чисел с фиксированной запятой: $[X]_{\text{пр}} = 1.01011$, $[Y]_{\text{пр}} = 0,00111$. Результат округлить до пятого знака после запятой.

15. Выполните умножение чисел с фиксированной запятой $[X]_{\text{пр}} = 0,11011$; $[Y]_{\text{пр}} = 1,10011$.

16. Выполните операцию деления чисел с фиксированной запятой $(x:y)$:

$$[X]_{\text{пр}} = 0.01100; [Y]_{\text{пр}} = 1.10100.$$

17. Выполните операцию деления чисел с фиксированной запятой $(x:y)$:

$$[X]_{\text{пр}} = 0.0011; [Y]_{\text{пр}} = 1.0100.$$

Глава 3. ЛОГИЧЕСКИЕ ОСНОВЫ ЭВМ

3.1. Двоичные переменные и переключательные функции

Для формального описания узлов ЭВМ при их анализе и синтезе используется аппарат алгебры логики. Основные положения алгебры логики разработал в XIX в. английский математик Джордж Буль. Алгебру логики называют также булевой алгеброй.

В булевой алгебре различают двоичные переменные и переключательные функции.

Двоичные переменные могут принимать два значения: 0 и 1. Они называются также логическими или булевыми переменными и обозначаются символами x_1, x_2, x_3, \dots .

Переключательные функции (ПФ) зависят от двоичных переменных. Они, как и аргументы, могут принимать лишь два значения: 0 или 1. ПФ называют также логическими или булевыми функциями. Будем обозначать ПФ в виде $f(x_1, x_2, \dots)$, указывая в скобках аргументы, либо в виде y_1, y_2, \dots . ПФ в свою очередь могут служить аргументами еще более сложных логических функций. Следовательно, можно построить ПФ любой заранее заданной сложности, пользуясь ограниченным числом логических связей.

ПФ принято задавать таблицами истинности, в которых для всех наборов переменных указываются соответствующие им значения ПФ. Формирование значений ПФ в таблице истинности выполняется в соответствии с логикой работы устройства (сумматора, сдвигателя, преобразователя кодов и т. д.).

Набор переменных — это совокупность значений двоичных переменных, каждая из которых может быть равна 0 или 1. Если число аргументов (независимых переменных) ПФ равно n (т. е. x_1, x_2, \dots, x_n), то существует 2^n различных сочетаний этих переменных, т. е. наборов.

Табл. 3.1 представляет собой таблицу истинности для некоторых ПФ f_1 и f_2 , зависящих от двоичных переменных x_1, x_2, x_3 . Так как $n = 3$ (три переменных), табл. 3.1 содержит 8 строк, соответствующих $2^3 = 8$ наборам пере-

менных x_1, x_2, x_3 . Для каждого набора в табл. 3.1 записаны значения ПФ f_1 и f_2 , равные 0 или 1. По таблице истинности записывается аналитическое выражение для ПФ (рассматривается в следующих параграфах).

Таблица 3.1

x_1	x_2	x_3	f_1	f_2
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

3.2. Элементарные логические функции

Произвольная ПФ может быть выражена в форме функции от двоичных переменных (либо от других ПФ) с помощью ограниченного числа элементарных логических функций. Рассмотрим эти функции.

Логическое отрицание (функция НЕ). Логическим отрицанием переменной x называется такая ПФ $f_1(x)$, которая имеет значение 1, когда $x = 0$ и значение 0, когда $x = 1$. ПФ НЕ обозначается в виде $f_1 = \bar{x}$ и читается: « f_1 есть (эквивалентно) не x ».

Табл. 3.2 представляет собой таблицу истинности логической функции НЕ.

Таблица 3.2

x	f_1
0	1
1	0

Функцию НЕ выполняет физический элемент (электронная схема), который называется элементом НЕ или инвертором. Обозначение инвертора на функциональных схемах показано на рис. 3.1.

Логическое умножение (конъюнкция). Конъюнкция двух (или любого другого числа) переменных x_1 и x_2 принимает значение 1 только на наборе, в котором все пе-

ременные имеют значения 1. На остальных наборах эта функция имеет значение 0.

Таблица 3.3

x_1	x_2	f_2
0	0	0
0	1	0
1	0	0
1	1	1

Табл. 3.3 представляет собой таблицу истинности конъюнкции двух переменных x_1 и x_2 . ПФ конъюнкция обозначается в виде

$$f_2 = x_1 x_2$$

и читается: « f_2 есть (эквивалентно) x_1 и x_2 ».

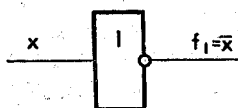


Рис. 3.1

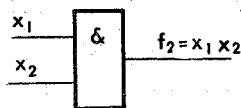


Рис. 3.2

Для обозначения конъюнкции можно использовать символы \wedge или $\&$. Конъюнкция называется также функцией И, так как она имеет значение 1, только если первый и второй аргументы имеют значения 1.

Функция И выполняется электронной схемой, которая называется элементом И или конъюнктором. Обозначение элемента И на функциональных схемах показано на рис. 3.2. Число входов элемента И равно числу переменных, участвующих в операции умножения.

Логическое сложение (дизъюнкция). Дизъюнкция двух (или любого другого числа) переменных x_1 и x_2 имеет значение 0 только на наборе, в котором все переменные имеют значение 0. Если хотя бы одна из переменных равна 1, функция будет иметь значение 1.

Табл. 3.4 есть таблица истинности для дизъюнкции двух переменных x_1 и x_2 . ПФ дизъюнкция записывается в виде

$$f_3 = x_1 + x_2$$

и читается: « f_3 есть (эквивалентно) x_1 или x_2 ». Кроме символа $+$, для дизъюнкции употребляется символ \vee .

Так как функция дизъюнкции имеет значение 1, если первый или второй аргументы имеют значение 1, операция дизъюнкции называется также операцией ИЛИ.

Таблица 3.4

x_1	x_2	f_3
0	0	0
0	1	1
1	0	1
1	1	1

Операция ИЛИ реализуется электронной схемой, которая называется элементом ИЛИ или дизъюнктом. Обозначение элемента ИЛИ на функциональных схемах показано на рис. 3.3. Число входов элемента ИЛИ равно числу переменных, участвующих в операции дизъюнкции.

Элементарные логические функции НЕ, И, ИЛИ являются основными логическими функциями. Имеется еще несколько логических функций, производных от основных функций (т. е. выражающихся через функции НЕ, И, ИЛИ), которые реализуются соответствующими электронными элементами и так часто встречаются в схемотехнике ЭВМ, что им были даны собственные названия. Рассмотрим эти функции.

Отрицание конъюнкции (операция И — НЕ). Эта функция образуется путем отрицания результата, получаемого при выполнении операции И. Табл. 3.5 есть таблица истинности операции И — НЕ для двух переменных. Из сравнения табл. 3.3 и 3.5 видно, что ПФ И — НЕ является отрицанием (операцией НЕ) конъюнкции. ПФ И — НЕ записывается в виде

$$f_4 = \overline{x_1 x_2}.$$

Таблица 3.5

x_1	x_2	f_4
0	0	1
0	1	1
1	0	1
1	1	0

Функцию И — НЕ выполняет схема, которая называется элементом И — НЕ. Обозначение элемента И — НЕ

на функциональных схемах показано на рис. 3.4. Число входов элемента И — НЕ определяется числом аргументов функции И — НЕ.



Рис. 3.3

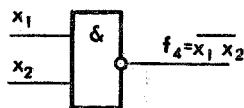


Рис. 3.4

Отрицание дизъюнкции (операция ИЛИ — НЕ). Эта операция образуется путем отрицания результата, полученного при выполнении операции ИЛИ. Табл. 3.6

Т а б л и ц а 3.6

x_1	x_2	f_5
0	0	1
0	1	0
1	0	0
1	1	0

представляет собой таблицу истинности операции ИЛИ — НЕ для двух переменных. Из сравнения табл. 3.4 и 3.6 видно, что ПФ ИЛИ — НЕ является отрицанием (операцией НЕ) дизъюнкции. ПФ ИЛИ — НЕ записывается в виде

$$f_5 = \overline{x_1 + x_2}.$$

Операцию ИЛИ — НЕ выполняет электронный элемент, который называется элементом ИЛИ — НЕ. Обозначение элемента ИЛИ — НЕ на функциональных схемах показано на рис. 3.5. Число входов элемента ИЛИ — НЕ определяется числом аргументов функции ИЛИ — НЕ.

ИСКЛЮЧАЮЩЕЕ ИЛИ (операция НЕРАВНОЗНАЧНОСТЬ или СЛОЖЕНИЕ ПО МОДУЛЮ ДВА). Данная функция имеет значение 1 на тех наборах переменных, в которых число единиц нечетно. Для двух переменных операция НЕРАВНОЗНАЧНОСТЬ иллюстрируется таблицей истинности (табл. 3.7). Эта операция записывается для двух переменных в виде

$$f_6 = x_1 \oplus x_2.$$

Условное обозначение элемента, выполняющего функцию НЕРАВНОЗНАЧНОСТЬ, на функциональных схемах приведено на рис. 3.6. Символ M2 в поле элемента означает «сложение по модулю два».

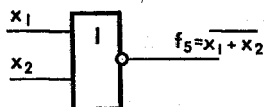


Рис. 3.5

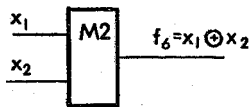


Рис. 3.6

Операция НЕРАВНОЗНАЧНОСТЬ выражается через операции НЕ, И, ИЛИ в виде

$$f_6 = x_1 \bar{x}_2 + \bar{x}_1 x_2.$$

Таблица 3.7

x_1	x_2	f_6
0	0	0
0	1	1
1	0	1
1	1	0

Операция ИСКЛЮЧАЮЩЕЕ ИЛИ — НЕ (РАВНОЗНАЧНОСТЬ). Функция РАВНОЗНАЧНОСТЬ представляет собой отрицание операции ИСКЛЮЧАЮЩЕЕ ИЛИ.

Данная операция имеет значение 1 на тех наборах переменных, которые содержат четное число единиц. Для двух переменных операция ИСКЛЮЧАЮЩЕЕ ИЛИ — НЕ представлена таблицей истинности (табл. 3.8). Эта операция записывается для двух переменных в виде

$$f_7 = x_1 \oplus x_2.$$

Таблица 3.8

x_1	x_2	f_7
0	0	1
0	1	0
1	0	0
1	1	1

Операция ИСКЛЮЧАЮЩЕЕ ИЛИ — НЕ выражается через операции НЕ, И, ИЛИ в виде

$$f_7 = \overline{x_1 x_2} + x_1 x_2.$$

Функцию РАВНОЗНАЧНОСТЬ выполняет электронный элемент с аналогичным названием, изображение которого на функциональных схемах дано на рис. 3.7.

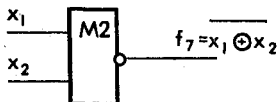


Рис. 3.7

3.3. Законы булевой алгебры

В булевой алгебре используются четыре основных закона: переместительный, сочетательный, распределительный, инверсии. Эти законы позволяют проводить эквивалентные преобразования ПФ, записанных с помощью операций НЕ, И, ИЛИ, т. е. приводить выражения ПФ к удобному (более простому) виду. Рассмотрим эти законы.

Переместительный закон аналогичен переместительному закону обычной алгебры и записывается в виде:

а) для дизъюнкции

$$x_1 + x_2 = x_2 + x_1; \quad (3.1)$$

б) для конъюнкции

$$x_1 x_2 = x_2 x_1. \quad (3.2)$$

Таким образом, от перемены мест слагаемых (с множителей) их логическая сумма (логическое произведение) не меняется.

Сочетательный закон также аналогичен сочетательному закону обычной алгебры и записывается в виде:

а) для дизъюнкции

$$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3; \quad (3.3)$$

б) для конъюнкции

$$x_1 (x_2 x_3) = (x_1 x_2) x_3. \quad (3.4)$$

Следовательно, можно группировать переменные, объединенные знаком дизъюнкции или конъюнкции, это не меняет значений ПФ.

Распределительный закон записывается в виде:

а) для дизъюнкции

$$x_1 + x_2x_3 = (x_1 + x_2)(x_1 + x_3), \quad (3.5)$$

т. е. дизъюнкция переменной и конъюнкции равносильна конъюнкции дизъюнкций этой переменной с сомножителями;

б) для конъюнкции

$$x_1(x_2 + x_3) = x_1x_2 + x_1x_3, \quad (3.6)$$

т. е. конъюнкция переменной и дизъюнкции эквивалентна дизъюнкции конъюнкций этой переменной со слагаемыми.

Справедливость выражения (3.5) доказывается путем составления таблиц истинности для левой и правой частей. Значения этих таблиц совпадают для одинаковых наборов переменных, это и доказывает справедливость (3.5).

Закон инверсии:

а) для дизъюнкции

$$\overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2, \quad (3.7)$$

т. е. отрицание дизъюнкции логических переменных эквивалентно конъюнкции отрицаний этих переменных;

б) для конъюнкции

$$\overline{x_1x_2} = \bar{x}_1 + \bar{x}_2, \quad (3.8)$$

т. е. отрицание конъюнкции переменных эквивалентно дизъюнкции отрицаний этих переменных.

Справедливость выражений (3.7) и (3.8), как и (3.5), доказывается также путем составления таблиц истинности для левой и правой частей каждого выражения и их сравнения на совпадение для одних и тех же наборов переменных.

Докажем, например, справедливость формулы (3.8). Для этого составим таблицу истинности (табл. 3.9) правой и левой части соотношения (3.8).

Из табл. 3.9 видно, что содержимое столбцов 4 и 7 совпадает на одних и тех же наборах переменных (т. е. в одних и тех же строках). Это и доказывает справедливость формулы (3.8).

Из законов алгебры логики выводится ряд важных правил, которые полезны при выполнении эквивалентных преобразований ПФ.

Таблица 3.9

x_1	x_2	$x_1 x_2$	$\overline{x_1 x_2}$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_1 + x_2}$
1	2	3	4	5	6	7
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

1. Выражения, имеющие всегда значение 1:

$$x + 1 = 1;$$

$$x + \overline{x} = 1.$$

2. Выражения, имеющие всегда значение 0:

$$x \cdot 0 = 0;$$

$$x \cdot \overline{x} = 0.$$

3. Двойное отрицание: $\overline{\overline{x}} = x$.

4. Повторение:

$$x \cdot x \dots x = x;$$

$$x + x + \dots + x = x.$$

5. Склеивание: $x_1 x_2 + x_1 \overline{x_2} = x_1$.

Это правило доказывается путем вынесения в левой части x_1 за скобку и применения правила 1 для выражения, имеющего значение 1:

$$x_1 x_2 + x_1 \overline{x_2} = x_1 (x_2 + \overline{x_2}) = x_1 \cdot 1 = x_1.$$

6. Поглощение:

$$x_1 + x_1 x_2 = x_1.$$

Для доказательства справедливости этого правила вынесем в левой части за скобку x_1 и применим правило 1 для выражения, имеющего значение 1:

$$x_1 + x_1 x_2 = x_1 (1 + x_2) = x_1 \cdot 1 = x_1.$$

3.4. Представление переключательных функций

ПФ могут быть выражены различными логическими формулами благодаря возможности проведения над ними эквивалентных преобразований. Однако на практике наиболее удобными для представления ПФ оказываются дизъюнктивные и конъюнктивные формы.

Рассмотрим вначале дизъюнктивные формы представления ПФ, среди которых различают

дизъюнктивную нормальную форму и совершенную дизъюнктивную нормальную форму. В основе представления ПФ в дизъюнктивных формах лежит понятие *элементарной конъюнкции*.

Конъюнкция любого числа двоичных переменных x_1, x_2, \dots, x_n называется элементарной, если сомножителями в ней являются либо одиночные аргументы, либо отрицания одиночных аргументов. Например, конъюнкции $\overline{x_1 x_2 x_4}$, $\overline{x_1 \overline{x_2} x_3}$ являются элементарными, а конъюнкции $x_2 x_3$, $x_1 \overline{x_2} x_3$ ими не являются. Очевидно, что символ любой переменной в элементарной конъюнкции может встречаться только один раз, так как произведение переменной на себя равно этой же переменной (по правилу повторения), а произведение переменной на свое отрицание равно нулю.

Дизъюнктивной нормальной формой (ДНФ) переключательной функции называется дизъюнкция (логическая сумма) любого числа элементарных конъюнкций. Например, ПФ

$$f = \overline{x_1 x_2 x_3} + \overline{x_2 x_4} + \overline{x_2} + \overline{x_3 x_4}$$

записана в ДНФ, так как она представляет собой логическую сумму элементарных конъюнкций.

Число переменных, входящих в элементарную конъюнкцию, определяет *ранг* этой конъюнкции. Например, $\overline{x_1}$, $\overline{x_2}$, $\overline{x_3}$ — конъюнкции 1-го ранга; $\overline{x_2 x_3}$, $\overline{x_1 x_4}$ — конъюнкции 2-го ранга и т. д.

Совершенной ДНФ (СДНФ) ПФ, имеющей n аргументов, называется такая форма, в которой все конъюнкции имеют ранг n .

СДНФ переключательной функции записывается по таблице истинности. Для того чтобы по таблице истинности построить СДНФ, надо каждому набору переменных, на котором ПФ принимает единичное значение, поставить в соответствие элементарную конъюнкцию ранга n и все эти конъюнкции соединить дизъюнктивно; в каждой конъюнкции СДНФ с отрицанием берутся те переменные, которые на соответствующем этой конъюнкции наборе имеют нулевое значение.

Используем это правило и запишем в СДНФ функции f_1 и f_2 трех переменных, заданных табл. 3.1. В результате получим следующие выражения:

$$f_1 = \overline{x_1 \overline{x_2} x_3} + \overline{x_1 x_2 x_3} + \overline{x_1 \overline{x_2} x_3}; \quad (3.9)$$

$$f_2 = \overline{x_1 x_2 x_3} + \overline{x_1 x_2 x_3} + \overline{x_1 x_2 x_3} + \overline{x_1 x_2 x_3}. \quad (3.10)$$

Элементарные конъюнкции СДНФ называют *конституентами* (составляющими) *единицы*, так как они соответствуют наборам, при которых ПФ принимает значения 1.

После составления выражения для ПФ можно построить из логических элементов И, ИЛИ, НЕ функциональную схему, которая будет вычислять значение ПФ.

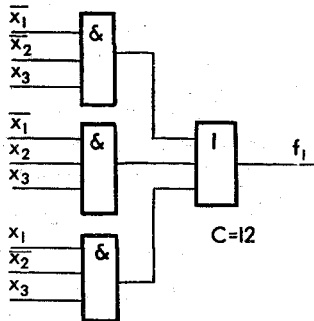


Рис. 3.8

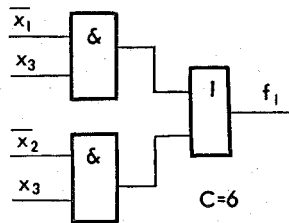


Рис. 3.9

Чтобы построить функциональную схему для ПФ, записанной в СДНФ, надо взять k схем (элементов) И на n входов каждая, где k — число конъюнкций СДНФ, и объединить выходы схем И с помощью элемента ИЛИ на k входов. Для ПФ (формула (3.9)) функциональная (или логическая) схема приведена на рис. 3.8 (на схеме не показаны лишь элементы НЕ, с помощью которых получают инверсные значения переменных x_1 и x_2).

Для каждой функциональной схемы можно сделать оценку ее сложности, которая выражается *ценой* (по Квайну) схемы C . Цена C определяется суммарным числом входов логических элементов. Чем меньше величина C , тем проще функциональная схема. Если ПФ задана в СДНФ, ее цену можно выразить формулой

$$C = kn + k.$$

Используя правило склеивания, можно упростить ПФ, заданную в СДНФ. Для этого в СДНФ сначала склеиваются между собой конъюнкции ранга n , затем полученные конъюнкции ранга $(n - 1)$, $(n - 2)$, и так до тех пор, пока в выражении для ПФ не останется ни одной пары склеиваемых между собой конъюнкций. Операция склеивания позволяет понизить ранг конъюнкций и сократить их число. В результате уменьшается цена функциональной схемы. Например, в выражении (3.9) выполним склеива-

ние конъюнкций $\overline{x_1x_2x_3}$ и $\overline{x_1x_2x_3}$ по переменной x_2 и конъюнкций $\overline{x_1x_2x_3}$ и $x_1x_2x_3$ по переменной x_1 . В результате функция f_1 преобразуется к виду

$$f_1 = \overline{x_1}x_3 + \overline{x_2}x_3. \quad (3.11)$$

Функциональная схема, реализующая ПФ f_1 по выражению (3.11), изображена на рис. 3.9. Цена этой схемы $C = 6$, т. е. по сравнению с эквивалентной схемой на рис. 3.8 цена уменьшилась в 2 раза.

В результате попарного склеивания конъюнкций ранга n , а далее $(n - 1)$, $(n - 2)$ и т. д., в выражении для ПФ остаются конъюнкции, которые между собой не склеиваются. Конъюнкция, которая не склеивается ни с какой другой конъюнкцией ДНФ, называется *простой импликантой*. В выражении (3.11) конъюнкции $\overline{x_1}x_3$ и $\overline{x_2}x_3$ не склеиваются между собой, т. е. они являются простыми импликантами. ДНФ, представляющая собой дизъюнкцию простых импликант, называется *сокращенной ДНФ*.

Конъюнктивные формы представления ПФ используются реже, чем дизъюнктивные. Здесь также различают конъюнктивную нормальную и совершенную конъюнктивную нормальную формы. Аналогично элементарной конъюнкции определяется *элементарная дизъюнкция*, представляющая собой дизъюнкцию любого числа двоичных переменных x_1, x_2, \dots, x_n , в которой слагаемыми являются либо одиночные аргументы, либо отрицания одиночных аргументов. Например, дизъюнкции $\overline{x_2} + x_3, x_1 + \overline{x_2} + x_3, \overline{x_2}$ являются элементарными, а $x_2 + x_3, x_1 + x_2 + x_3$ ими не являются. Как и в элементарной конъюнкции, символ любой переменной в элементарной дизъюнкции может встречаться только один раз.

Конъюнктивной нормальной формой (КНФ) ПФ называется конъюнкция (логическое произведение) элементарных дизъюнкций.

Например, функции

$$f_1 = (x_2 + \overline{x_3})(x_1 + \overline{x_2})(x_2 + \overline{x_3} + x_4)\overline{x_1}$$

и

$$f_2 = x_2 = (x_1 + \overline{x_2})(\overline{x_2} + x_3 + \overline{x_4})$$

заданы в КНФ. Ранг элементарной дизъюнкции определяется числом переменных, входящих в эту дизъюнкцию.

Совершенной КНФ (СКНФ) ПФ, имеющей n аргументов, называется такая форма, в которой все дизъюнкции имеют ранг n . СКНФ переключательной функции

составляется по таблице истинности по наборам, при которых ПФ принимает нулевые значения. Каждому такому набору ставится в соответствие элементарная дизъюнкция ранга n , и все эти дизъюнкции соединяются символами конъюнкции. При этом в дизъюнциях с отрицанием берутся те переменные, которые на соответствующем наборе имеют единичное значение.

Для функций f_1 и f_2 , заданных в табл. 3.1, можно записать следующие СКНФ:

$$f_1 = (x_1 + x_2 + x_3)(x_1 + \bar{x}_2 + x_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3); \quad (3.12)$$

$$f_2 = (x_1 + x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + x_2 + \bar{x}_3). \quad (3.13)$$

Элементарные дизъюнкции СКНФ называют *конституентами нуля*, так как они соответствуют наборам, на которых ПФ принимает значение 0.

По аналогии с ДНФ для КНФ можно дать и другие определения. Однако вследствие того, что для упрощения ПФ чаще используются дизъюнктивные формы, мы ограничимся данной о КНФ информацией.

3.5. Функционально полные системы логических функций

Любую переключательную функцию f двоичных переменных x_1, x_2, \dots, x_n можно вычислить путем использования логических операций И, ИЛИ и НЕ. С помощью электронных элементов И, ИЛИ и НЕ можно построить функциональную схему, реализующую любую переключательную функцию.

Набор элементарных логических функций является *функционально полным*, если с его помощью можно записать в виде формулы любую переключательную функцию. Очевидно, что набор функций {И, ИЛИ, НЕ} является функционально полным. Набор физических (электронных) элементов, реализующих эти три операции, образует функционально полную систему логических элементов.

Свойствами функциональной полноты обладают также наборы {И, НЕ} и {ИЛИ, НЕ}. Действительно, в наборе {И, НЕ} функцию ИЛИ можно выполнить через операции И, НЕ путем эквивалентного преобразования $f = x_1 + x_2 = x_1 + x_2 = \overline{\overline{x_1} \cdot \overline{x_2}}$; в наборе {ИЛИ, НЕ} функцию И можно выполнить через операции ИЛИ и НЕ следующим

образом: $f = x_1 x_2 = \overline{\overline{x_1 x_2}} = \overline{x_1 + x_2}$. Таким образом, набор функций {И, ИЛИ, НЕ} обладает избыточностью.

Функционально полными являются функции И — НЕ, ИЛИ — НЕ. Имея необходимое количество логических элементов И — НЕ либо ИЛИ — НЕ с требуемым числом входов, можно реализовать (т. е. построить функциональную схему) любую переключательную функцию.

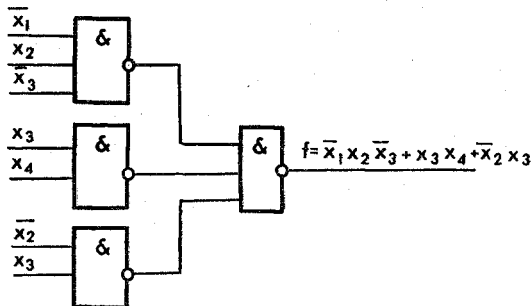


Рис. 3.10

Для вычисления переключательной функции путем выполнения операции И — НЕ ее необходимо вначале представить в ДНФ. Выполнив далее двойное отрицание и применив закон отрицания, мы приведем ПФ к требуемому виду. Например, ПФ $f = x_1 x_2 x_3 + x_3 x_4 +$

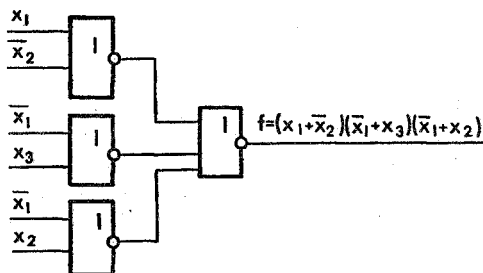


Рис. 3.11

$+ x_2 x_3$ в соответствии с описанными действиями приводится к виду

$$f = \overline{\overline{x_1 x_2 x_3 + x_3 x_4 + x_2 x_3}} = \overline{x_1 x_2 x_3 \cdot x_3 x_4 \cdot x_2 x_3}. \quad (3.14)$$

По выражению (3.14) ПФ реализуется на элементах И — НЕ (в базисе И — НЕ) в соответствии с рис. 3.10.

Для реализации ПФ в базисе ИЛИ — НЕ необходимо

записать ее сначала в КНФ. Выполнив далее двойное отрицание и применив закон отрицания, мы получим искомую форму. Например, ПФ $f = (x_1 + \bar{x}_2)(\bar{x}_1 + x_3) \times \times (\bar{x}_1 + x_2)$ в соответствии с этим принимает вид

$$\begin{aligned} f &= \overline{(x_1 + \bar{x}_2)(\bar{x}_1 + x_3)(\bar{x}_1 + x_2)} = \\ &= \overline{(x_1 + \bar{x}_2)} + \overline{(\bar{x}_1 + x_3)} + \overline{(\bar{x}_1 + x_2)}. \end{aligned} \quad (3.15)$$

Выражение (3.15) реализуется на элементах ИЛИ — НЕ в соответствии с рис. 3.11.

3.6. Минимизация переключательных функций

Минимизация, или сокращение выражения, для ПФ необходима, чтобы обеспечить минимум затрат оборудования при построении функциональной схемы в заданном базисе (наборе) логических элементов. Чаще всего используется набор И, ИЛИ, НЕ, а минимизация ПФ ведется в ДНФ. В рамках нормальных форм минимальной будет такая разновидность функции, которая состоит из наименьшего количества дизъюнктивных (конъюнктивных) членов при наименьшем суммарном числе символов переменных. В случае ДНФ в результате минимизации получается минимальная ДНФ (МДНФ).

Для минимизации ПФ используют два основных метода: 1) метод Квайна; 2) метод карт Карно (диаграмм Вейча).

Метод Квайна применяется к функциям, заданным в СДНФ (возможно задание и в СКНФ), и проводится в два этапа.

На *первом этапе* выполняется переход от СДНФ к сокращенной ДНФ путем проведения всех возможных склеиваний друг с другом сначала конъюнкций ранга n , затем ранга $n - 1$, далее $n - 2$ и т. д. Каждый раз в группе конъюнкций ранга r ($1 \leq r \leq n$) отыскиваются пары конъюнкций вида Ax и $A\bar{x}$, где A — общая часть этих конъюнкций. Эти конъюнкции склеиваются между собой по переменной x (например, конъюнкций $x_1x_2x_3\bar{x}_4$, $x_1x_2x_3x_4$ имеют общую часть $A = x_1x_2x_3$ и склеиваются по переменной x_4). При этом получается конъюнкция A ранга $r - 1$, а конъюнкции Ax и $A\bar{x}$ помечаются и сравниваются также со всеми другими конъюнкциями ранга r с целью выполнения операции склеивания. Результа-

том выполнения последовательности попарного сравнения и склеивания конъюнкций ранга r является группа конъюнкций ранга $r-1$ и непомеченные конъюнкции ранга r . Непомеченные конъюнкции ранга r не участвовали в склеивании, следовательно, являются простыми импликантами и включаются в сокращенную ДНФ. Конъюнкции ранга $r-1$ вновь подвергаются попарному сравнению и склеиванию, как это было описано выше для конъюнкций ранга r ; в результате имеем группу конъюнкций ранга $r-2$ и непомеченные конъюнкции ранга $r-1$, которые являются простыми импликантами, включаемыми далее в сокращенную ДНФ. Первый этап заканчивается тогда, когда вновь полученная группа конъюнкций не содержит склеивающихся членов, т. е. содержит только простые импликанты. После этого записывается сокращенная ДНФ ПФ, в которую включаются все полученные простые импликанты.

Рассмотрим реализацию первого этапа на примере переключательной функции

$$f = \overline{x_1 x_2 x_3} + \overline{x_1 x_2 x_3} + \overline{x_1 x_2 x_3} + \overline{x_1 x_2 x_3} + \overline{x_1 x_2 x_3}, \quad (3.16)$$

где для удобства последующего изложения над конъюнкциями записаны присвоенные им номера.

Анализируя в выражении (3.16) всевозможные пары конъюнкций 1—2, 1—3, ..., 1—5, 2—3, ..., 2—5, 3—4, 3—5, 4—5, находим, что операция склеивания выполняется между парами 1—3, 2—5, 3—4, 4—5. Таким образом, все конъюнкции исходной СДНФ участвуют в склеивании и помечаются подчеркиванием.

Исходная СДНФ (3.16) записывается в виде

$$f = \overline{x_1 x_3} + \overline{x_2 x_3} + \overline{x_1 x_2} + \overline{x_1 x_3}. \quad (3.17)$$

Продолжая для выражения (3.17) процедуру склеивания, сравниваем попарно конъюнкции 1—2, 1—3, 1—4, 2—3, 2—4 и 3—4. Эти пары конъюнкций между собой не склеиваются. Поэтому полученная запись ПФ (3.17) представляет собой сокращенную ДНФ исходной ПФ (3.16), содержащей только простые импликанты.

Второй этап заключается в переходе от сокращенной ДНФ к тупиковым ДНФ и выборе среди них МДНФ. *Тупиковой* называется такая ДНФ, среди простых импликант которой нет ни одной лишней. При этом под лишней понимается такая простая импликанта, удаление которой

не влияет на значение истинности этой функции. Возможны случаи, когда в сокращенной форме нет лишних простых импликант. Тогда сокращенная ДНФ является тупиковой.

Для выявления лишних простых импликант строится импликантная матрица, которая называется также матрицей (таблицей) покрытий. Каждая строка импликантной матрицы соответствует одной простой импликанте, а столбцы — конституентам единицы, которыми они и помечаются. Для рассматриваемого примера импликантная матрица приведена в табл. 3.10.

Таблица 3.10

Простые импликанты	Конституенты единицы				
	$\bar{x}_1\bar{x}_2\bar{x}_3$	$\bar{x}_1x_2x_3$	$x_1\bar{x}_2\bar{x}_3$	$x_1\bar{x}_2x_3$	$x_1x_2x_3$
1 x_2x_3	×		×		
2 x_2x_3		×			×
3 x_1x_2			×	×	
4 x_1x_3			×	×	×

Нахождение тупиковых ДНФ по импликантной матрице начинается с разметки матрицы. При этом каждая импликанта сравнивается со всеми конституентами единицы. Если импликанта является собственной частью некоторой конституенты, то на пересечении строки и столбца ставится условный знак, например \times . Конституенты единицы, помеченные в строке с простой импликантой, поглощаются (покрываются) этой простой импликантой. Это значит, что на соответствующих наборах данная импликанта обеспечивает единичные значения ПФ. Например, простая импликанта x_2x_3 , входящая в сокращенную ДНФ (3.17) рассматриваемого примера, является собственной частью конституент единиц $x_1x_2x_3$ и $x_1x_2x_3$, и поэтому в строке с данной простой импликантой (табл. 3.10) поставлены две пометки \times в соответствующих колонках. Аналогично размечены другие строки табл. 3.10.

Выявление лишних простых импликант выполняется

следующим образом. В импликантной таблице условно вычеркивается строка с проверяемой простой импликантой вместе с соответствующими пометками в строке. Если при этом окажется, что в каждом столбце импликантной таблицы остается хотя бы по одной пометке, проверяемая импликанта является лишней и ее следует удалить. Оставшиеся простые импликанты покрывают все единичные значения ПФ. Испытание каждой последующей простой импликанты возможно лишь после удаления уже выявленных лишних простых импликант. Изменение последовательности испытаний и удаления лишних членов сокращенной ДНФ может привести к различным тупиковым формам ПФ, из которых выбирается МДНФ.

Из табл. 3.10 видно, что только простая импликанта 1 обеспечивает единичное значение ПФ на наборе 000, а импликанта 2 — на наборе 011 (соответствующие этим наборам столбцы обозначены конstituентами единицы $\bar{x}_1\bar{x}_2x_3$ и $x_1x_2x_3$), поэтому эти простые импликанты обязательно войдут во все тупиковые ДНФ. Простую импликанту 3 можно считать лишней. Однако, если ее сохранить, лишней можно считать простую импликанту 4. Таким образом, для ПФ возможны две тупиковые формы:

$$f_{1T} = \bar{x}_2\bar{x}_3 + x_2x_3 + x_1x_3, \quad (3.18)$$

в которую не включена лишняя импликанта $x_1\bar{x}_2$, и

$$f_{2T} = \bar{x}_2\bar{x}_3 + x_2x_3 + x_1\bar{x}_2, \quad (3.19)$$

в которую не вошла лишняя импликанта x_1x_3 .

Тупиковые формы (3.18) и (3.19) имеют одинаковое суммарное количество переменных, поэтому любую из них можно выбрать в качестве МДНФ. Например, можно считать $f_{мин} = f_{1T} = \bar{x}_2\bar{x}_3 + x_2x_3 + x_1x_3$. Эта ПФ реализуется схемой с ценой $C = 9$.

Метод диаграмм Вейча (карт Карно) удобен для минимизации ПФ, содержащих обычно не более четырех переменных.

Диаграмма Вейча имеет вид прямоугольника (квадрата), разбитого на 2^n клеток, где n — число аргументов ПФ. Каждой клетке диаграммы ставится в соответствие определенная конъюнкция, причем конъюнкции располагаются таким образом, чтобы в соседних клетках (в строке или в столбце) они отличались не более чем значением одной переменной. В результате любые две соседние

в строке или в столбце конъюнкции склеиваются по соответствующей переменной. Соседними на диаграмме являются также крайние (левая и правая) конъюнкции в одной строке и (нижняя и верхняя) конъюнкции в одном столбце.

Диаграммы Вейча для двух, трех и четырех переменных изображены на рис. 3.12, *а*, *б*, *в* соответственно.

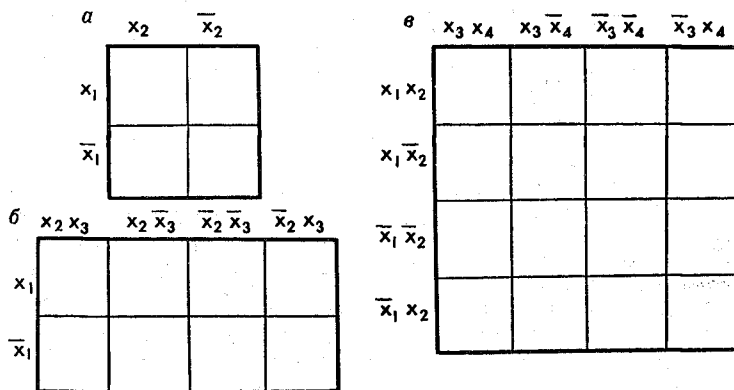


Рис. 3.12

Для минимизации ПФ приводится к СДНФ, после чего заполняется диаграмма Вейча для n переменных. При этом в соответствующую клетку диаграммы вписывается 1, если ПФ на данном наборе аргументов равна единице. В остальные клетки вписывают нули либо такие клетки вообще оставляют пустыми.

В заполненной диаграмме обводят прямоугольными контурами клетки с единицами, после чего записывают МДНФ ПФ в виде дизъюнкции простых импликант, описывающих эти контуры. При проведении контуров придерживаются следующих правил: внутри контура должны быть только клетки с единицами; количество клеток с единицами должно выражаться величиной 2^i (где $i = 0, 1, 2, \dots$), т. е. может быть равно 1, 2, 4, 8 и т. д.; единицы в крайних клетках одного столбца или одной строки могут включаться в один контур; каждый контур должен включать как можно большее число клеток с единицами, а общее число контуров должно быть как можно меньшим.

В простую импликанту, описывающую контур, включаются те переменные, которые во всех клетках контура

имеют либо только прямое, либо только инверсное значение.

Рассмотрим примеры минимизации ПФ с помощью диаграмм Вейча.

Пример 3.1. Минимизировать ПФ $f_1 = \bar{x}_1 x_2 + x_1 \bar{x}_2 + \bar{x}_1 \bar{x}_2$.

Заполняем диаграмму Вейча для двух переменных (рис. 3.13). Далее проводим два контура, охватывающих клетки с единицами.

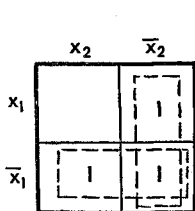


Рис. 3.13

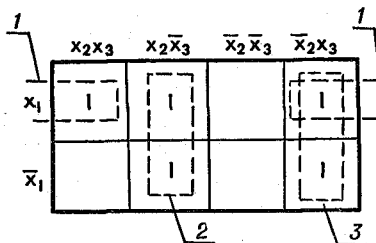


Рис. 3.14

Для нахождения простой импликаты, описывающей контур, надо выяснить, от каких переменных не зависит данный контур. Например, вертикальный контур на рис. 3.13 охватывает строки с x_1 и \bar{x}_1 , следовательно, в простую импликанту, описывающую этот контур, переменная x_1 входить не будет. Аналогично, в простую импликанту, описывающую горизонтальный контур, переменная x_2 входить не будет. Следовательно, ПФ f_1 будет иметь следующую МДНФ: $f_{1 \text{ мин}} = \bar{x}_1 + x_2$.

Пример 3.2. Минимизировать ПФ $f_2 = x_1 x_2 + x_1 \bar{x}_2 x_3 + \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 x_3$. Приводим ПФ к СДНФ: $f_2 = x_1 x_2 (x_3 + \bar{x}_3) + \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 x_3 = x_1 x_2 x_3 + x_1 x_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 x_3$. Заполняем диаграмму Вейча и проводим 3 контура (рис. 3.14):

$$f_{2 \text{ мин}} = x_1 x_3 + x_2 \bar{x}_3 + \bar{x}_2 x_3.$$

В этом примере вместо контура 1 можно было бы провести контур, охватывающий конъюнкции $x_1 x_2 x_3$ и $x_1 x_2 \bar{x}_3$. Тогда МДНФ f_2 имела бы вид: $f_{2 \text{ мин}} = x_1 x_2 + x_2 \bar{x}_3 + \bar{x}_2 x_3$. Таким образом, для ПФ f_2 возможны две тупиковые (минимальные) формы, равноценные по числу букв и слагаемых, и каждую из них можно выбрать для построения комбинационной схемы.

Пример 3.3. Минимизировать ПФ

$$f_3 = x_1 x_2 x_3 x_4 + \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + x_1 x_2 \bar{x}_3 x_4 + \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 x_3 x_4 + \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 \bar{x}_3 x_4.$$

Заполняем диаграмму Вейча на 4 переменные и проводим 4 контура (рис. 3.15). МДНФ f_3 запишется в виде

$$f_{3 \text{ мин}} = x_2 x_4 + x_2 \bar{x}_3 + x_1 \bar{x}_3 x_4 + \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4.$$

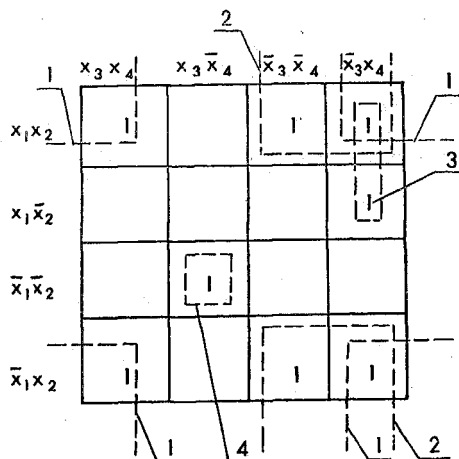


Рис. 3.15

3.7. Основные сведения о конечном автомате

В ЭВМ преобразование информации выполняется логическими схемами, которые подразделяются на два класса:

- 1) комбинационные схемы, или автоматы без памяти;
- 2) последовательностные схемы, или автоматы с памятью.

Синтез комбинационных (или логических) схем был рассмотрен в предыдущих параграфах данной главы. Каждому набору (комбинации) входных сигналов $x_1, x_2, \dots, x_l, \dots, x_L$ комбинационной схемы соответствует определенная комбинация выходных сигналов $y_1, y_2, \dots, y_n, \dots, y_N$, являющаяся выходной функцией (L — число входов комбинационной схемы, N — число ее выходов).

Последовательностная схема состоит из логических и запоминающих элементов. В качестве запоминающих элементов используются триггеры (рассматриваются в следующей главе). Значения выходных сигналов последовательностных схем зависят как от текущих значений входных сигналов, так и от значений входных сигналов, поступавших на схему в предыдущие моменты (такты) времени.

Используются две модели цифровых автоматов с памятью: абстрактная и структурная, в соответствии

с которыми автомат называется абстрактным либо структурным. *Абстрактная модель* применяется при теоретическом рассмотрении автоматов. *Структурная модель* служит для построения схемы автомата из логических элементов и триггеров, которая выполняет функцию устройства управления.

Абстрактным автоматом называют дискретный преобразователь информации с конечным входным алфавитом $Z = \{z_1, \dots, z_f, \dots, z_F\}$, конечным выходным алфавитом $W = \{w_1, \dots, w_g, \dots, w_G\}$, конечным множеством внутренних состояний $A = \{a_1, \dots, a_m, \dots, a_M\}$ и двумя характеристическими функциями: функцией переходов δ и функцией выходов λ .

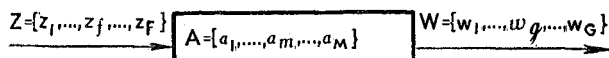


Рис. 3.16

Алфавит есть конечное множество попарно различных символов, которые называются буквами этого алфавита. Алфавит, как любое множество, задается перечислением его элементов, т. е. символов. Примеры алфавитов: $A = \{\alpha, \beta, \gamma, *, \neq\}$; $B = \{x, y, z\}$.

Абстрактный автомат имеет один входной и один выходной каналы (рис. 3.16). Функционирование автомата происходит в дискретные моменты автоматного времени, ход которого обозначается натуральными числами $t = 0, 1, 2, \dots$.

В каждый момент дискретного времени t автомат находится в определенном состоянии $a(t) = a_m$, воспринимает на входном канале некоторую букву входного алфавита $z(t) = z_f$ (входной сигнал), выдает на выходном канале некоторую букву выходного алфавита $w(t) = w_g$ (выходной сигнал), определяемую функцией выходов λ как $w(t) = \lambda(a(t), z(t))$ или $w_g = \lambda(a_m, z_f)$, и переключается в новое состояние $a(t+1) = a_s$, которое определяется функцией переходов δ как $a(t+1) = \delta(a(t), z(t))$ или $a_s = \delta(a_m, z_f)$.

Абстрактный автомат называется *конечным*, так как множества A, Z, W конечны. Кроме того, он называется *полностью определенным*, если для любой пары (a_m, z_f) определены функции δ и λ . У частичного автомата функции δ и λ определены не для всех пар (a_m, z_f) . При рассмотрении функционирования автомата считается, что

исходным состоянием в момент $t=0$ является $a(0) = a_1$, которое называется *начальным состоянием*.

Закон функционирования автоматов может задаваться в виде уравнений, таблиц и графов. Под законом функционирования понимается совокупность правил, описывающих последовательность переключения состояний автомата и последовательность выходных сигналов в зависимости от последовательности входных сигналов.

В зависимости от способа определения значений выходных сигналов различают два типа автоматов.

Автоматами первого типа или автоматами Мура называют такие автоматы, в которых выходной символ $w(t)$ не зависит явно от входного символа $z(t)$, а определяется внутренним состоянием автомата в момент времени t . Иначе говоря, автоматы Мура описываются системой уравнений:

$$\begin{aligned} a(t+1) &= \delta(a(t), z(t)); \\ w(t) &= \lambda(a(t)). \end{aligned}$$

Другой тип автоматов составляют автоматы Мили, у которых функции переходов и выходов описываются следующей системой уравнений:

$$\begin{aligned} a(t+1) &= \delta(a(t), z(t)); \\ w(t) &= \lambda(a(t), z(t)). \end{aligned}$$

Таким образом, в автомате Мили выходной сигнал (буква выходного алфавита) в момент автоматного времени t зависит как от внутреннего состояния автомата в момент t , так и от входного сигнала (буквы входного алфавита) в момент t .

При табличном задании закона функционирования автомата Мили используют таблицы переходов и таблицы выходов.

В табл. 3.11 (таблица переходов) и 3.12 (таблица выходов) приведен пример задания автомата Мили, у ко-

Т а б л и ц а 3.11

$z(t) \backslash a(t)$	a_1	a_2	a_3	a_4
z_1	a_1	a_3	a_3	a_3
z_2	a_4	a_1	a_4	a_2
z_3	a_2	a_3	a_4	a_1

Таблица 3.12

$z(t) \backslash a(t)$	a_1	a_2	a_3	a_4
z_1	w_1	w_1	w_2	w_1
z_2	w_2	w_2	w_3	w_1
z_3	w_1	w_1	w_2	w_3

торого $A = \{a_1, a_2, a_3, a_4\}$; $Z = \{z_1, z_2, z_3\}$; $W = \{w_1, w_2, w_3\}$. Строки таблиц отмечены входными сигналами, а столбцы — состояниями. Крайний левый столбец таблиц отмечается начальным состоянием автомата a_1 . Входные сигналы и состояния, отмечающие строки и столбцы таблиц, относятся к моменту времени t , т. е. отражают $z(t)$ и $a(t)$. На пересечении столбца $a(t) = a_m$ и строки $z(t) = z_f$ в таблице переходов указывается состояние $a(t+1) = a_s$, определяемое функцией переходов $a_s = \delta(a_m, z_f)$. В состоянии a_s автомат переключается из состояния a_m под действием сигнала z_f . В таблице выходов на пересечении столбца $a(t) = a_m$ и строки $z(t) = z_f$ ставится соответствующий этому переходу выходной сигнал $w(t) = w_g$, определяемый функцией выходов $w_g = \lambda(a_m, z_f)$.

По табл. 3.11 и 3.12 можно проследить последовательность работы автомата. В начальный момент времени $t=0$ автомат находится в исходном состоянии $a(0) = a_1$. Если на входном канале в момент $t=0$ будет действовать, к примеру, буква $z(0) = z_2$, то автомат сформирует на выходном канале букву $w_2 = w(0) = \lambda(a(0), z(0))$ и в следующий момент времени $t=1$ переключится в новое состояние $a_4 = a(1) = \delta(a(0), z(0))$.

В момент времени $t=1$ автомат, находясь в состоянии $a(1) = a_4$, может воспринять на входном канале любую букву из множества Z , например $z(1) = z_2$. Тогда автомат сформирует на выходном канале букву $w_1 = w(1) = \lambda(a(1), z(1))$ и в следующий момент времени $t=2$ переключится в новое состояние $a_2 = a(2) = \delta(a(1), z(1))$ и т. д. Таким образом, если на вход автомата, предварительно установленного в начальное состояние a_1 , подавать в последовательные моменты времени $t=0, 1, 2, \dots$ некоторую последовательность букв входного алфавита $z(0), z(1), z(2), \dots$ (входное слово), то на выходе автомата будут последовательно появляться буквы выходного алфавита $w(0), w(1), w(2), \dots$ (выходное слово); при этом автомат будет переключаться в состояния $a(1), a(2),$

$a(3), \dots$. Выходное слово называется также реакцией автомата на входное слово.

Идентичность структур табл. 3.11 и 3.12 позволяет объединить их в одну так называемую совмещенную таблицу переходов и выходов (табл. 3.13). С помощью таблиц переходов и выходов можно найти выходную реакцию автомата на любое входное слово.

Таблица 3.13

$z(t) \backslash a(t)$	a_1	a_2	a_3	a_4
z_1	a_1 w_1	a_1 w_1	a_3 w_2	a_3 w_1
z_2	a_4 w_2	a_1 w_2	a_4 w_3	a_2 w_1
z_3	a_2 w_1	a_3 w_1	a_4 w_2	a_1 w_3

Если функции переходов и выходов автомата определены не на всех упорядоченных парах символов (a_m, z_f) , то автомат не полностью определен, поэтому некоторые клетки таблиц переходов и выходов (или совмещенной таблицы переходов и выходов) будут пусты. В этом случае в клетках таблиц ставятся прочерки. Табл. 3.14 и 3.15 являются таблицами соответственно переходов и выходов частичного автомата Мили.

Таблица 3.14

$z(t) \backslash a(t)$	a_1	a_2	a_3
z_1	a_2	a_3	—
z_2	a_1	—	a_1

Таблица 3.15

$z(t) \backslash a(t)$	a_1	a_2	a_3
z_1	w_2	w_2	—
z_2	w_1	—	w_3

Поскольку в автомате Мура выходной сигнал не зависит явно от входного сигнала, а только от состояния, то автомат Мура задается отмеченной таблицей переходов (табл. 3.16), в которой каждый столбец отмечается состоянием $a(t) = a_m$ и выходным сигналом $w(t) = w_g = \lambda(a_m)$, соответствующим этому состоянию.

Другим, более наглядным и в некоторых случаях более удобным способом задания закона функционирования автомата является представление его в виде графа. *Граф автомата* изображается в виде совокупности вершин,

Таблица 3.16

$z(t)$	w_3	w_1	w_2	w_2
	a_1	a_2	a_3	a_4
z_1	a_2	a_4	a_4	a_3
z_2	a_3	a_2	a_3	a_3
z_3	a_4	a_1	a_2	a_2

каждой из которых ставится в соответствие одно из внутренних состояний автомата, и дуг, соединяющих вершины графа и соответствующих переходам между

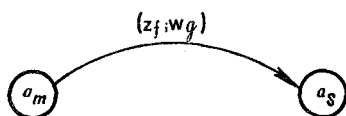


Рис. 3.17

состояниями. Вершины изображаются кружочками, а дуги — стрелками. Направленность графа, которая отображается дугами, позволяет проследить последовательность переключений состояний автомата при подаче на его входной канал последовательности букв входного алфавита. В каждую вершину графа вписывается символ внутреннего состояния, которому соответствует данная вершина.

Если в автомате Мили под действием входной буквы z_f осуществляется переход из состояния a_m в состояние a_s с выдачей выходной буквы w_g , то вершины графа a_m (исходное состояние) и a_s (состояние перехода) соединяются дугой, направленной от a_m к a_s , а дуга отмечается парой символов $(z_f; w_g)$, как это показано на рис. 3.17.

В общем случае переход автомата из состояния a_m в состояние a_s происходит под действием нескольких входных символов; при каждом таком переходе формируется свой выходной символ. Очевидно, что в этом случае следует провести столько дуг из a_m в a_s , сколько имеется переходов между ними, причем каждая дуга должна отмечаться парой символов из Z и W .

Чтобы не загромождать изображение автомата графом, множество дуг, исходящих из вершины a_m и входящих в вершину a_s , заменяют одной дугой, отмеченной дизъюнкцией пар символов, которыми размечаются исходные дуги. Например, в графе автомата Мили, заданного табл. 3.13, от a_3 к a_4 должны идти две дуги

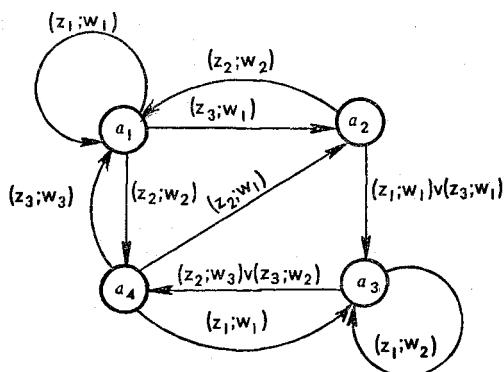


Рис. 3.18

так как в автомате из состояния a_3 возможен переход в a_4 под действием входной буквы z_2 с выдачей выходной буквы w_3 либо под действием входной буквы z_3 с выдачей выходной буквы w_2 . Обе дуги графа можно заменить одной, отмеченной дизъюнкцией вида $(z_2; w_3) \vee (z_3; w_2)$. Точно так же одной дугой можно изобразить переходы из a_2 в a_3 . Прделав это, получим граф автомата Мили, изображенный на рис. 3.18.

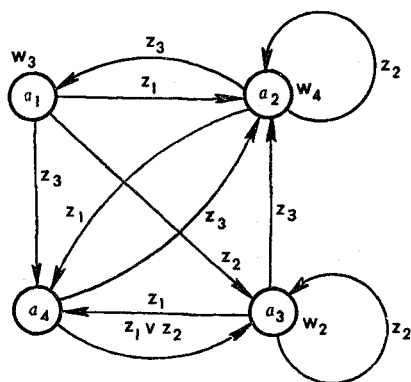


Рис. 3.19

При описании автомата Мура в виде графа выходной сигнал w_g записывается внутри вершины a_m или рядом с ней, а дуга, соответствующая переходу из a_m в a_s , отмечается входным сигналом z_f , под действием которого происходит этот переход. Если имеется множество дуг, исходящих из a_m и входящих в a_s , то они заменяются одной дугой, которую отмечают дизъюнкцией входных сигналов. Граф автомата Мура, ранее заданного отмеченной таблицей переходов 3.16, приведен на рис. 3.19.

Контрольные вопросы и задания

1. Запишите функцию $f = x_1 + \bar{x}_1 x_2$ в СДНФ.
2. Используя операцию склеивания, преобразуйте ПФ $f = x_1 x_2 x_3 + x_1 x_2 \bar{x}_3$. По какой переменной произошло склеивание конъюнкций?
3. Используя операцию поглощения, преобразуйте ПФ $f = x_1 \bar{x}_2 + x_1 x_2 x_3 x_4$.
4. Используя закон инверсии, преобразуйте ПФ $f = \overline{x_1 + x_2 + x_3 + x_4}$. Какая форма ПФ получилась?
5. Для функции $f = x_1 \bar{x}_2 \bar{x}_3 + x_1 \bar{x}_2 x_3 + x_1 x_2 \bar{x}_3 + x_1 x_2 x_3$ запишите сокращенную ДНФ.
6. Запишите функцию $f = x_1 x_2 + x_2 \bar{x}_3 x_4$ в форме, удобной для реализации на элементах И — НЕ.
7. Запишите функцию $f = (x_1 + x_2)(\bar{x}_2 + \bar{x}_3 + x_4)$ в форме, удобной для реализации на элементах ИЛИ — НЕ.
8. Найдите минимальную форму ПФ $f = x_1 x_2 \bar{x}_3 + x_1 x_2 x_3 + x_1 \bar{x}_2 x_3 + x_1 \bar{x}_2 \bar{x}_3 + x_1 x_2 \bar{x}_3$, воспользовавшись методом Квайна либо карт Карно.
9. Чем отличается автомат Мили от автомата Мура?

Глава 4. СХЕМОТЕХНИЧЕСКИЕ ОСНОВЫ ЭВМ

4.1. Базовые элементы ЭВМ

Общие сведения. Схемотехника — научно-техническое направление, занимающееся проблемами анализа и синтеза* отдельных приборов радиотехники, связи, автоматики, вычислительной техники с целью обеспечения оптимального выполнения ими заданных функций и расчета параметров входящих в них элементов.

Элементами электронных вычислительных машин называются наименьшие функциональные части, на которые можно разделить машину при ее логическом проектировании и технической реализации. Элементы ЭВМ выполняют хранение, преобразование и передачу двоичных переменных, а также ряд вспомогательных функций: задержку сигнала во времени, формирование сигнала с определенными физическими характеристиками и т. п.

Основу элементной базы ЭВМ составляют цифровые интегральные схемы (ИС). Их сложность характеризуется степенью функциональной интеграции $K_{\text{и}}$ ($K_{\text{и}} = \lg N_{\text{эл}}$, где $N_{\text{эл}}$ — число элементов И — НЕ либо ИЛИ — НЕ, расположенных на кристалле ИС).

В зависимости от сложности интегральные схемы подразделяются на следующие типы:

МИС — *малые интегральные схемы*, содержащие один или несколько логических элементов ($K_{\text{и}} \approx 1$);

СИС — *средние интегральные схемы*, содержащие один или несколько функциональных узлов (сумматоров, счетчиков и др.) и имеющие обычно $K_{\text{и}} = 1 \dots 2$;

БИС — *большие интегральные схемы*, содержащие одно или несколько функциональных устройств (арифметическое устройство, арифметико-логическое устройство АЛУ, устройство управления УУ, запоминающее устройство ЗУ) и имеющие $K_{\text{и}} = 2 - 4$;

* *Анализ* — исследование работы известной схемы, *синтез* — создание новой схемы.

СБИС — *сверхбольшие БИС*, выполняющие функции целых цифровых систем (например, микроЭВМ) и имеющие $K_{\text{и}} > 4$.

По типу обрабатываемых сигналов элементы ЭВМ подразделяются на *потенциальные, импульсные* и *импульсно-потенциальные*. Длительность потенциального сигнала не ограничена сверху, как правило,

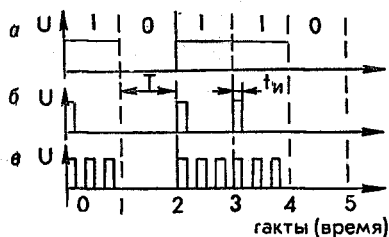


Рис. 4.1

кратна длительности такта T и определяется частотой смены информации. Длительность импульсного сигнала имеет некоторое постоянное стандартное значение $t_n < T$ и не зависит от частоты смены информации. Типы сигналов определяют три способа представления (кодирования) информации: потенциальный (рис. 4.1, а), импульсный (рис. 4.1, б) и динамический (рис. 4.1, в).

По функциональному назначению элементы делятся на *логические, элементы памяти* и *специальные*. К логическим относят элементы, реализующие функции из базисной системы (базисную функцию). Элементы памяти (триггеры) предназначены для запоминания (хранения) информации, а специальные элементы — для физического преобразования электрических сигналов, т. е. усиления, формирования, генерирования, задержки и т. п.

Основные характеристики и параметры логических элементов (ЛЭ). К основным статическим характеристикам элементов ЭВМ относятся *входная* $I_{\text{вх}} = f(U_{\text{вх}})$, *выходная* $I_{\text{вых}} = f(U_{\text{вых}})$ и *амплитудная передаточная* (АПХ) $U_{\text{вых}} = f(U_{\text{вх}})$ характеристики. В статических характеристиках отсутствует время t в качестве аргумента функции.

К статическим параметрам относятся *входные* и *выходные напряжения* логических 0 и 1 ($U_{\text{вых}}^0$, $U_{\text{вых}}^1$, $U_{\text{вх}}^0$, $U_{\text{вх}}^1$), *входные* и *выходные токи* логических 0

и 1 ($I_{вх}^0, I_{вх}^1, I_{вых}^0, I_{вых}^1$). Один уровень напряжения, например высокий $U^1 \geq 2,4$ В, в двоичных элементах принимается за единичный, а другой, например низкий $U^0 \leq 0,4$ В, — за нулевой. Если $U^1 > U^0$, то говорят о положительной логике (ПЛ), если $U^1 < U^0$ — об отрицательной (ОЛ).

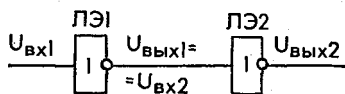


Рис. 4.2

Кроме того, к статическим параметрам относятся *логический перепад* $U_{\text{л}} = U^1 - U^0$ и *средняя* (статическая) *потребляемая мощность* $P_{\text{ср}} = 0,5(P^0 + P^1)$, где P^0 — мощность, потребляемая элементом от источника питания при $U_{\text{вых}} = U_{\text{вых}}^0$; P^1 — мощность, потребляемая элементом при $U_{\text{вых}} = U_{\text{вых}}^1$.

ЛЭ должны обладать помехоустойчивостью, т. е. нечувствительностью к действию помех определенной величины. *Статическую помехоустойчивость* определяют по АПХ. На рис. 4.2 показана последовательная цепочка инвертирующих элементов. Пример изменения входного напряжения $U_{\text{вх}2}$ ЛЭ2 от времени t под действием помех при одинаковых АПХ элементов показан на рис. 4.3.

Если под действием положительной помехи $U_{\text{вх}2}$ увеличится больше порогового напряжения нуля ЛЭ2 $U_{\text{пор}2}^0$, то выходное напряжение $U_{\text{вых}2}$ начнет уменьшаться, что может привести к ложному срабатыванию элементов, подключенных к выходу ЛЭ2. На участке $U_{\text{пор}}^0 < U_{\text{вх}} < U_{\text{пор}}^1$ дифференциальный коэффициент усиления

$\left| \frac{dU_{\text{вых}}}{dU_{\text{вх}}} \right| > 1$ и триггерные схемы могут переключаться в ложное состояние

$$\left| \frac{dU_{\text{вых}}}{dU_{\text{вх}}} \right|_{U_{\text{вх}}=U_{\text{пор}}^0} = \left| \frac{dU_{\text{вых}}}{dU_{\text{вх}}} \right|_{U_{\text{вх}}=U_{\text{пор}}^1} = 1.$$

Статическая помехоустойчивость* определяется в вольтах и равна максимально допустимой амплитуде

* В справочниках приводится значение помехоустойчивости, определенное для семейства предельных передаточных характеристик, полученных при различных температурах.

помехи, т. е. помехоустойчивость U_n^+ к положительным помехам (или помехоустойчивость по уровню логического 0 $U_{\text{пом}}^0 = U_n^+$) определяется из выражения

$$U_n^+ = U_{\text{пор}}^0 - U^0, \quad (4.1)$$

а помехоустойчивость к отрицательным помехам U_n^- (или что то же помехоустойчивость по уровню логической 1 $U_{\text{пом}}^1 = U_n^-$)

$$U_n^- = U^1 - U_{\text{пор}}^1. \quad (4.2)$$

Статическими помехами принято считать помехи, значение которых остается постоянным в течение времени, превышающего длительность этапов переходного процесса ЛЭ.

К динамическим параметрам относятся параметры, зависящие от времени: *время задержки распространения сигнала* при включении $t_{\text{зд.р}}^{1,0}$ (интервал времени между входным и выходным импульсами при переходе напряжения на выходе ИС от напряжения 1

к напряжению 0, измеренному на уровне 0,5 или на заданных значениях напряжения) и *время задержки распространения сигнала* при выключении $t_{\text{зд.р}}^{0,1}$. На рис. 4.4, а показан входной, а на рис. 4.4, б выходной сигнал инвертирующего элемента. Часто для характеристики быстродействия элементов используется средняя задержка распространения сигнала $t_{\text{зд.р.ср}} = 0,5(t_{\text{зд.р}}^{1,0} + t_{\text{зд.р}}^{0,1})$, измеряемая обычно в наносекундах ($1 \text{ нс} = 10^{-9} \text{ с}$).

Иногда указывается *время перехода* ИС из 1 в 0 $t^{1,0}$, а также время перехода из 0 в 1 $t^{0,1}$.

Динамическим параметром является также *динамическая мощность* $P_{\text{дин}}$, т. е. мощность, потребляемая от источника питания во время переключения из 0 в 1 и из 1 в 0.

Ряд параметров учитывает как статику, так и динамику ЛЭ:

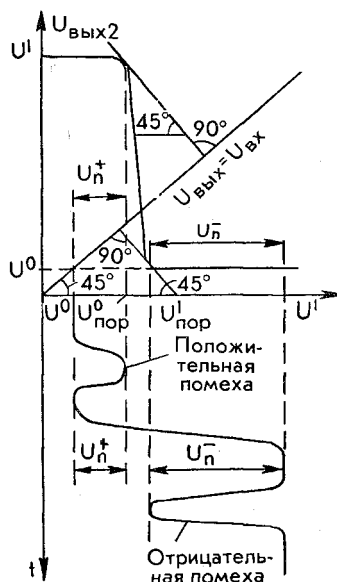


Рис. 4.3

коэффициент разветвления по выходу $K_{\text{раз}}$ — максимальное число единичных нагрузок, которое можно одновременно подключить к выходу ИС. Единичной нагрузкой является один вход основного ЛЭ данной серии ИС;

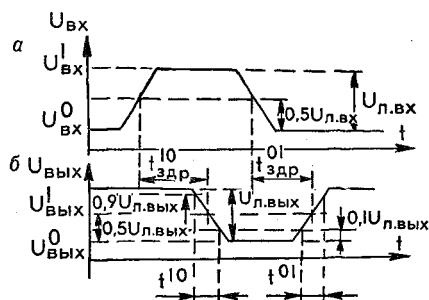


Рис. 4.4

коэффициент объединения по входу $K_{\text{об}}$ — число входов ИС, по которым реализуется логическая функция;

коэффициент объединения по выходу $K_{\text{об. вых}}$ — максимальное число соединяемых между собой выходов ИС;

потребляемая мощность (полная мощность) $P_{\text{пот}} = P_{\text{ср}} + P_{\text{дин}}$ — значение мощности, потребляемой от источника питания в заданном режиме (при заданных C_n, f_p).

Для сравнения различных серий ЛЭ между собой используется параметр A — энергия переключения, представляющая собой произведение: $P_{\text{пот}} \cdot t_{\text{зд.р.ср}}$. Те ЛЭ лучше, у которых параметр A меньше:

$$|A| = |P_{\text{пот}}| \cdot [t_{\text{зд.р.ср}}] = \text{мВт} \cdot \text{нс} = \frac{\text{мДж}}{\text{с}} \cdot \text{нс} = \text{пДж}.$$

Диодно-резисторные логические схемы (ДРЛ). Реальные вольт-амперные характеристики (ВАХ) кремниевого диода (а также перехода база — эмиттер кремниевого транзистора) и диода Шотки показаны на рис. 4.5, а.

Удобно пользоваться упрощенными аппроксимированными ВАХ диодов (рис. 4.5, б или в), где $U_{\text{д.з}}$ — напряжение запирающего диода, $U_{\text{д.о}}$ — падение напряжения на открытом диоде, $r_{\text{д}}$ — сопротивление открытого диода. Обычно $r_{\text{д}}$ равно десяткам ом (например $r_{\text{д}} = 40 \text{ Ом}$), напряжение запирающего диода Шотки $U_{\text{д.ш.з}} = 0,3 \text{ В}$, напряжение на открытом диоде Шотки $U_{\text{д.ш.о}} = 0,4 \text{ В}$,

напряжение запираания кремниевого диода, эмиттерного перехода транзистора $U_{д.з} = U_{э.з} = 0,6$ В, падение напряжения на открытом кремниевом диоде (эмиттерном переходе база — эмиттер транзистора) $U_{д.о} = U_{э.о} =$

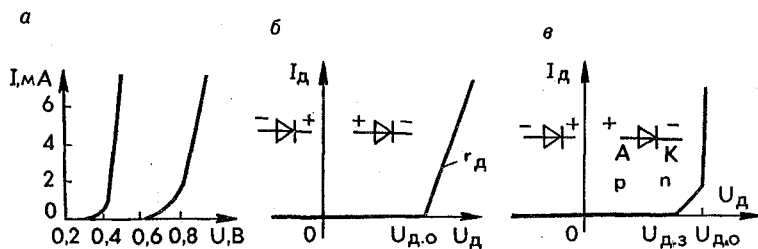


Рис. 4.5

$= (0,7 - 0,8)$ В. Напряжение $U_{д.ш.о}$ почти вдвое меньше $U_{д.о} = U_{э.о}$. Время выключения диода Шотки не превышает 0,1 нс, в то время как у полупроводниковых диодов это время составляет несколько десятых долей наносекунд, т. е. диод Шотки более быстродействующий.

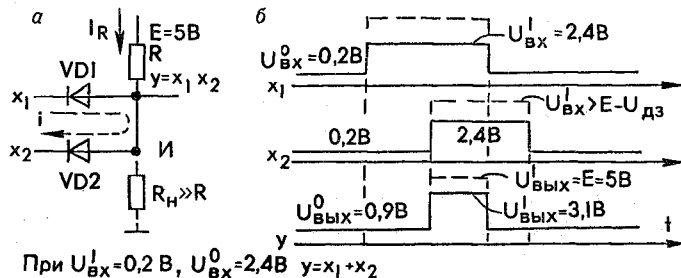


Рис. 4.6

Диод открыт, если напряжение на «анode А» (U_a) более положительное, чем на «катоде К» (U_k) на величину $U_{д.з}$ (рис. 4.5, в).

Диодно-резисторные схемы используются в диодно-транзисторной логике (ДТЛ), преобразователях уровней, постоянных запоминающих устройствах (ПЗУ), программируемых логических матрицах (ПЛМ) и др.

Рассмотрим диодно-резисторные логические схемы, входные и выходные напряжения, булевы функции в по-

положительной (ПЛ) и отрицательной (ОЛ) логике. Если сопротивление нагрузки $R_n \gg R$, то его можно не учитывать (на рис. 4.7, 4.8 не показано). Логический элемент ДРЛ на рис. 4.6, а выполняет функцию И в ПЛ

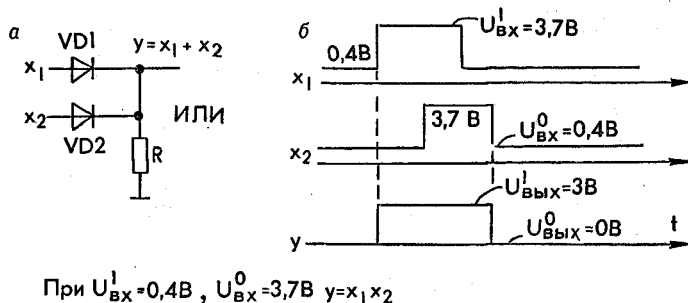


Рис. 4.7

(для высоких уровней, т. е. если за логическую 1 принять, например, 2,4 В, а за логический 0—0,2 В). Если на оба входа x_1 , x_2 поступают сигналы с уровнем

$$U_{вх} > E - U_{д.з} = 5 - 0,6 = 4,4 В,$$

(пунктир на рис. 4.6, б), то $U_{вых}^1 = 5 В$, так как $VD1$ и $VD2$ будут закрыты. ДРЛ на рис. 4.7, а является схемой ИЛИ на положительные сигналы (на высокие уровни). ДРЛ на рис. 4.8, а — это схема И на высокие уровни (в ПЛ) и схема ИЛИ на отрицательные сигналы (на низкие уровни). Соответственно ДРЛ на рис. 4.9, а в ПЛ является схемой ИЛИ на высокие уровни, а в ОЛ — схемой И на отрицательные сигналы (низкие уровни).

Транзисторные логические схемы с непосредственными (НСТЛ), резисторными (РСТЛ) и резисторно-конденсаторными (РКТЛ) связями. На рис. 4.10, а показана транзисторная логическая схема с непосредственными связями, а на рис. 4.10, б — с резисторными. Каждая из этих схем является простым инвертором, выполняющим булеву функцию $y = \bar{x}$. Сложным инвертором называется схема, содержащая несколько транзисторов.

Сравним эти схемы.

Основным недостатком НСТЛ является невысокая нагрузочная способность $K_{раз}$.

Для повышения $K_{\text{раз}}$ переходят к схемам РСТЛ. Как правило, РСТЛ содержит два резистора $R1, R2$ (рис. 4.10, б), позволяющих изменять $U_{\text{пор}}^0, U_{\text{пор}}^1$ на АПХ, т. е. увеличивать помехоустойчивость схемы.

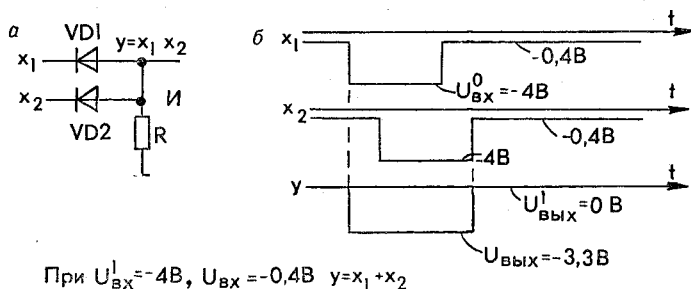


Рис. 4.8

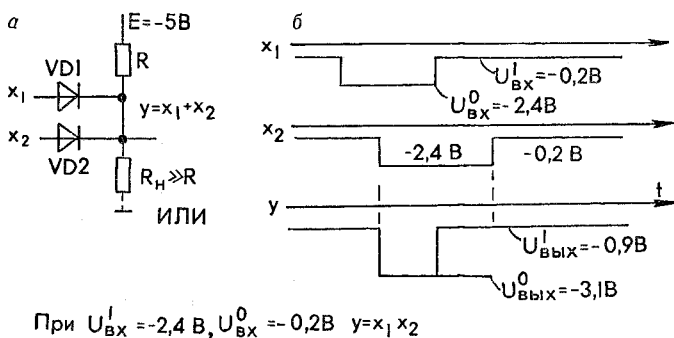


Рис. 4.9

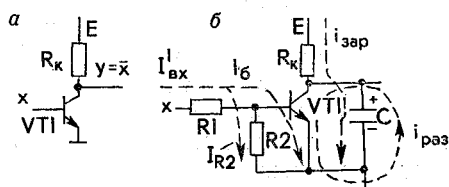


Рис. 4.10

Инверторы могут выполняться с открытым коллектором (ОК), т. е. без R_k . Открытый коллектор позволяет подключать различную нагрузку между коллектором

и источником питания E (лампочка накаливания, реле и т. п.), использовать более высоковольтный источник $E_1 > E$ для коллекторной цепи (при высоковольтном транзисторе), получать схемы с более высоким $I_{\text{вых}}$, а также объединять схемы по выходу для работы

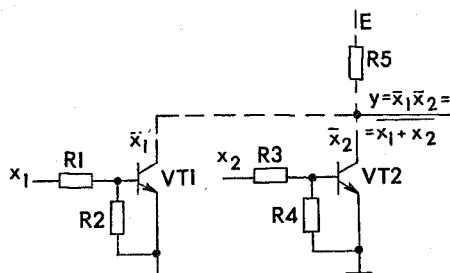


Рис. 4.11

на одну линию (рис. 4.11). При монтажном объединении коллекторов получается функция И:

$$y = \overline{x_1 x_2} = \overline{x_1} \overline{x_2} = x_1 + x_2.$$

На рис. 4.12 показана АПХ РСТЛ, приведенной на рис. 4.10, б. Когда напряжение на входе x достигает $U_{\text{пор}}$, транзистор открывается (на его базе напряжение достигает $U_{\text{э.з}} = 0,6 \text{ В}$). Поэтому

$$U_{\text{пор}}^0 R_2 / (R_1 + R_2) = U_{\text{э.з}},$$

откуда $U_{\text{пор}}^0 = U_{\text{э.з}} (R_2 + R_1) / R_2 = U_{\text{э.з}} (1 + R_1 / R_2)$.

Для ускорения включения и выключения транзистора, а следовательно, для увеличения быстродействия, в РСТЛ ставится форсирующий конденсатор $C_{\text{ф}}$, шунтирующий R_1 .

На рис. 4.13 показана временная диаграмма для РСТЛ (см. рис. 4.10, б) при работе на емкостную нагрузку $C_{\text{н}}$. В момент t_1 напряжение на входе x уменьшается до U^0 . Будем считать, что транзистор мгновенно закрывается. $C_{\text{н}}$ заряжается через $R_{\text{к}}$ током $i_{\text{зар}}$ от E за 3τ , где постоянная времени $\tau = R_{\text{к}} C_{\text{н}}$, т. е. $t_1^+ = 3R_{\text{к}} C_{\text{н}}$. В момент времени t_2 транзистор открывается и заряженный до напряжения питания E конденсатор $C_{\text{н}}$ разряжается током $i_{\text{раз}}$. Разрядка $C_{\text{н}}$ происходит постоянным коллекторным током $I_{\text{к}}$. Ток разрядки $i_{\text{раз}} = I_{\text{к}} = \beta I_{\text{б}}$, где β — коэффициент усиления тока базы транзистора;

I_6 — ток базы, который определяется, как: $I_6 = I_{вх} - I_{R2}$; $I_{вх}^1 = (U^1 - U_{э.о})/R1$; $I_{R2} = U_{э.о}/R2$. Так как C_n разряжается постоянным током I_K , то длительность отрицательного фронта t_{ϕ}^- определяется по формуле

$$t_{\phi}^- = EC_n / I_K.$$

Обычно $t_{\phi}^+ > t_{\phi}^-$, что является недостатком схемы простого инвертора. Для простого инвертора $R_{вык}^1 =$

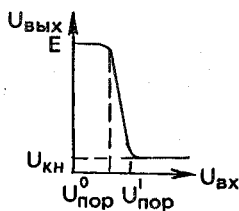


Рис. 4.12

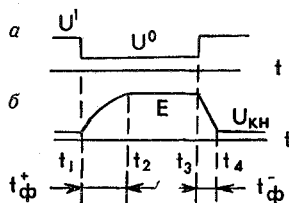


Рис. 4.13

$= R_K$, а $R_{вык}^0 = r_{к.н}$, где $r_{к.н}$ — сопротивление коллектора насыщенного транзистора.

Диодно-транзисторные логические схемы (ДТЛ). Базовый элемент ДТЛ, приведенный на рис. 4.14, состоит из ДРЛ И на высокие уровни ($VD1$, $VD2$, $R1$, см. рис. 4.6) и простого инвертора VT . Таблица истинности соответствует табл. 4.1.

Таблица 4.1

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

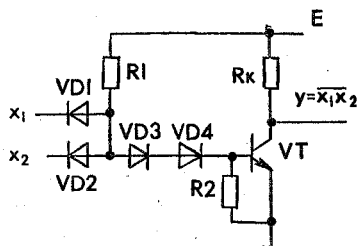


Рис. 4.14

Определим помехоустойчивость ДТЛ $U_{п.}^{\pm}$

Пусть $E = 5$ В, $U_{вх}^1 = 2,4$ В, $U_{вх}^0 = 0,2$ В.

При $x_2 = 1$, $x_1 = 0$ диод $VD1$ открыт, диоды $VD2$, $VD3$, $VD4$ и транзистор VT закрыты, на выходе $U_{вых}^1 = E$. Будем увеличивать напряжение на входе x_1 . Когда

напряжение на входе x_1 достигнет порогового напряжения $U_{\text{пор}}^0$, откроется транзистор VT , т. е. напряжение на его базе достигнет значения $U_{\text{э.з}} = 0,6$ В. Тогда $U_{\text{пор}}^0 = U_{\text{э.з}} + U_{\text{д.о.4}} + U_{\text{д.о.3}} - U_{\text{д.о.1}} \approx U_{\text{э.з}} + U_{\text{д.о}} = 0,6 + 0,7 = 1,3$ В, т. е. зависит от $VD3$, $VD4$; $U_{\text{пор}}^1 = U_{\text{э.о}} + U_{\text{д.о.4}} + U_{\text{д.о.3}} - U_{\text{д.о.1}} = U_{\text{э.о}} + U_{\text{д.о}} = 0,7 + 0,7 = 1,4$ В; $U_{\text{п}}^+ = U_{\text{пор}}^0 - U^0 = 1,3 - 0,2 = 0,9$ В; $U_{\text{п}}^- = U^1 - U_{\text{пор}}^1 = 2,4 - 1,4 = 1$ В.

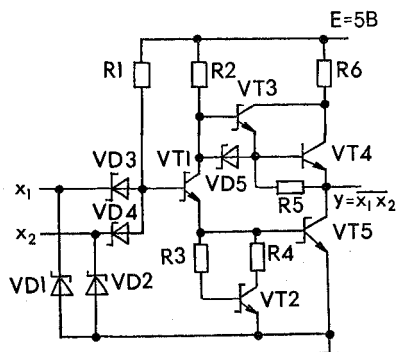


Рис. 4.15

Изменяя количество последовательно включенных диодов смещения $VD3$, $VD4$, можно менять $U_{\text{п}}^+$, $U_{\text{п}}^-$. На рис. 4.15 приведена схема базового элемента ДТЛ серии К555, имеющего параметры $t_{\text{зд.р}} = 10$ нс, $P_{\text{пот}} = 2$ мВт, $P_{\text{пот}} \cdot t_{\text{зд.р}} = 20$ пДж, $U_{\text{вых}}^1 \geq 2,7$ В, $U_{\text{вых}}^0 \leq 0,5$ В.

Схема состоит из ДРЛ И на высокие уровни ($VD3$, $VD4$, $R1$), выполненной на диодах Шотки, и сложного инвертора на транзисторе $VT4$ и транзисторах Шотки $VT1$, $VT2$, $VT3$, $VT5$.

Транзисторно-транзисторные логические схемы (ТТЛ).

На рис. 4.16 приведена схема базового элемента ТТЛ серии К155). Для данной серии $t_{\text{зд.р}} = 10$ нс, $P_{\text{пот}} = 10$ мВт, $P_{\text{пот}} \cdot t_{\text{зд.р}} = 100$ пДж, $U^1 \geq 2,4$ В, $U_{\text{п}}^{\pm} = 0,4$ В, $U^0 \leq 0,4$ В. В схеме (рис. 4.16) можно выделить 3 основные части: многоэмиттерный транзистор (МЭТ) $VT1$ с резистором $R1$, диоды $VD1$ и $VD2$, а также сложный инвертор ($VT2$, $VT2'$, $VT3$, $VT4$, $R2$, $R3'$, $R3''$, $R4$, $VD3$). Двухэмиттерный транзистор можно приближенно представить себе состоящим из двух транзисторов $n-p-n$ -типа (рис. (4.17)). МЭТ, выполняя функцию И (совместно с $R1$), занимает меньше места на кристалле, чем элементы

$VD1$, $VD2$, $VD3$, $R2$ в схеме на рис. 4.1, и, кроме того, он обладает усилительными свойствами. Демпфирующие диоды $VD1$, $VD2$ (рис. 4.16) служат для ограничения импульсов напряжения отрицательной полярности. Транзистор $VT2'$ предназначен для улучшения АПХ и повыше-

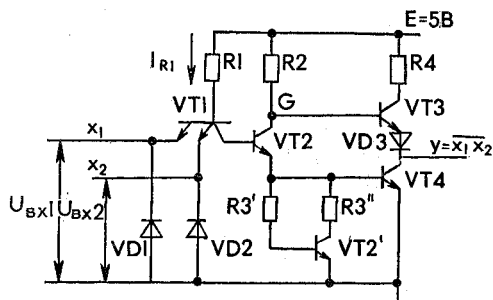


Рис. 4.16

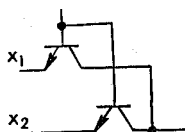


Рис. 4.17

ния $U_{п}^+$, а транзистор $VT2$ — для управления транзисторами $VT3$, $VT4$. Транзистор $VT4$ — это инвертор, аналогичный VT на рис. 4.14, а $VT3$ — эмиттерный повторитель, служащий для уменьшения $R_{вых}^1$. Диод $VD3$ обеспечивает запираание $VT3$ при открытом $VT4$.

Пусть $U_{Bx1} = U_{Bx2} = U^1 = 2,4$ В. Ток I_{R1} протекает от E через $R1$, переходы база — коллектор МЭТ, база — эмиттер $VT2$, база — эмиттер $VT4$ на «землю» (часть тока ответвляется также через $R3'$, $R3$, $VT2'$ на «землю»). Эмиттерные переходы $VT1$ смещены в обратном направлении. Транзистор $VT1$ работает в активном инверсном режиме; $VT2$, $VT2'$, $VT4$ открыты и насыщены $U_{вых}^0 = U_{к.н.б}^0 \approx 0,2$ В, $R_{вых}^0 = r_{к.н.4} = 5 - 20$ Ом.

Это же направление тока I_{R1} и значение напряжения на выходе $U_{вых} = U^0$ сохранится, если входы x_1 , x_2 будут свободны, т. е. не соединены ни с каким источником входного напряжения. Следовательно, свободный вход ТТЛ эквивалентен логической 1. Для увеличения быстродействия рекомендуется свободные входы подключать к U^1 , например к выходу инвертора, вход которого постоянно соединен с «землей». В серии 155 разрешается подключать до 20 свободных входов различных ИС к одному концу резистора сопротивлением 1 кОм, другой его конец подключается к источнику $E = 5$ В.

Рассмотрим другую комбинацию входных сигналов.

Пусть $U_{вх1} = 0,4$ В, $U_{вх2} = 2,4$ В. Ток I_{R1} протекает по цепи: E , $R1$, база — эмиттер $VT1$, вход x_1 , «земля». Транзисторы $VT2$, $VT2'$, $VT4$ закрываются, $VT1$ работает в режиме насыщения, напряжение $U_G \approx E = 5$ В, транзистор $VT3$ и диод $VD3$ открываются и на выходе

$$U_{\text{вых}}^1 = U_G^1 - U_{э.03} - U_{д.03} = 5 - 0,7 - 0,7 = 3,6 \text{ В.}$$

Рассмотренные ранее ЛЭ (рис. 4.10, а, б; рис. 4.14; рис. 4.15) имели 2 состояния выхода: U^0 либо U^1 . Логический элемент с ОК имеет также 2 состояния выхода: U^0 и состояние большого выходного сопротивления R_{off} , когда транзистор закрыт, что дает возможность объединять схемы по выходу для передачи информации по одной линии (рис. 4.11). Однако резистор $R5$ в схеме на рис. 4.11 не позволяет получать малые значения $t_{\text{ф}}^+$ при зарядке емкости нагрузки, подключенной к выходу, так как нельзя выбрать малым $R5$ из-за больших сквозных токов, протекающих по цепи: E , R_k , например открытый $VT2$, «земля». Эти сквозные токи увеличивают потребляемую мощность и уменьшают $K_{\text{раз}}$. Схемы ТТЛ (рис. 4.16) обеспечивают небольшие $t_{\text{ф}}^+$, но их нельзя объединять по выходу, так как будет протекать большой сквозной ток, который может привести к появлению на выходе y напряжения, не соответствующего уровням логического 0 и логической 1.

Можно получить элементы с тремя состояниями выхода (U^0 , U^1 , R_{off}) и малыми $R_{\text{вых}}^0$, $R_{\text{вых}}^1$. Вариантов схем с третьим состоянием много, но все они сводятся к подаче близкого к «земле» напряжения в точку G соединения коллектора $VT2$ с базой $VT3$, например с помощью инвертора РТЛ на транзисторе $VT5$ (рис. 4.18). Если $x_2 = 0$, $VT5$ закрыт и схема работает как обычный инвертор ТТЛ, т. е. $y = \bar{x}_1$. Если $x_2 = 1$, $VT5$ открывается и входит в насыщение. Малое напряжение $U_{к.н.5} = 0,2$ В приведет к запираанию сразу двух выходных транзисторов $VT3$, $VT4$, диода $VD3$ и появлению третьего состояния выхода с высоким выходным сопротивлением R_{off} .

Для обозначения третьего состояния выхода используется специальная метка \diamond .

Чтобы повысить быстродействие ТТЛ, используются транзисторы и диоды Шоттки. Транзистор Шотки можно рассматривать как обычный транзистор $n - p - n$ -типа, у которого $t_{\text{расс}} = 0,6 = 100 - 150$, $U_{к.н.ш} = (0,3 - 0,4)$ В, малый инверсный коэффициент усиления.

На рис. 4.19 приведена схема ЛЭ, реализующая булеву функцию И — ИЛИ — НЕ ($2И — 2ИЛИ — НЕ$).

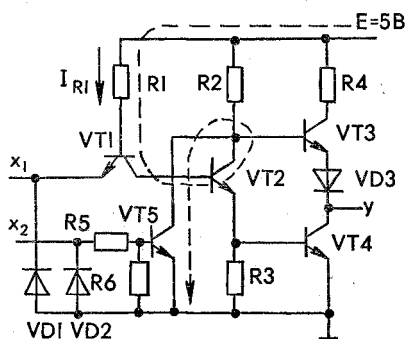


Рис. 4.18

Эмиттерно-связанные логические схемы (ЭСЛ). Быстродействие элементов во многом определяется активным режимом работы транзисторов и длительностью перезарядки емкостей. Из формулы зарядки (разрядки) емкости C на величину логического перепада $U_{\text{л}} = U^1 - U^0$ некоторым постоянным током зарядки $t_{\text{зар}} = CU_{\text{л}}/I_{\text{зар}}$ следует, что снизить $t_{\text{зар}}$ возможно, уменьшая $U_{\text{л}}$ и увеличивая $I_{\text{зар}}$. Эти принципы (активный режим, малый логический перепад и большие токи $I_{\text{зар}}$) реали-

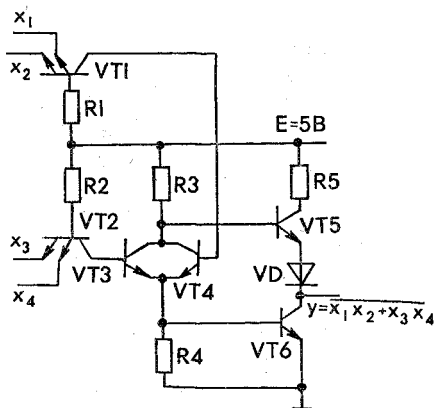


Рис. 4.19

зованы в самых быстродействующих ЛЭ ЭСЛ. Например, ЭСЛ серии К1500 имеет $t_{зд.р} < 1$ нс.

На рис. 4.20 приведена схема базового элемента ЭСЛ серии 500 с параметрами $t_{зд.р} = 1,5 - 2$ нс, $P_{пот} =$

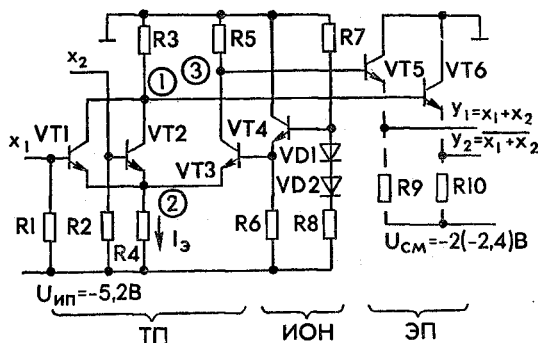


Рис. 4.20

$= 25$ мВт (без $R9, R10$), $U^1 = -(0,81 - 0,96)$ В $\approx -0,9$ В, $U^0 = (1,65 - 1,85)$ В $\approx -1,7$ В, $U_{ном}^0 = U_{п}^+ \geq 0,155$ В, $U_{ном}^1 = U_{п}^- \geq 0,125$ В, $U_{л} = 0,8$ В. Схема реализует на выходах функции 2ИЛИ/2ИЛИ — НЕ $y_1 = x_1 + x_2, y_2 = \underline{x_1 + x_2}$ в ПЛ ($U^1 > U^0$) либо 2И/2И — НЕ $y_1 = x_1 x_2, y_2 = \underline{x_1 x_2}$ в ОЛ ($U^1 < U^0$).

Данная схема состоит из токового переключателя ТП ($VT1, VT2, VT3, R1 - R5$), источника опорного напряжения ИОН ($VT4, R6 - R8, VD1, VD2$), эмиттерных повторителей ЭП ($VT5, VT6, R9, R10$). ИОН вырабатывает постоянное опорное напряжение $U_{оп}$, которое выбирается из условия $U_{оп} = (U^1 + U^0)/2 = [-0,9 + +(-1,7)]/2 = -1,3$ В и подается на базу $VT3$.

Рассмотрим две комбинации входных сигналов. Пусть $U_{вх1} = -0,9$ В $> U_{оп} = -1,3$ В $> U_{вх2} = -1,7$ В. Следовательно, $VT1$ откроется и будет работать в активном режиме. Для схемы ЭСЛ серии 500 $U_{э.01} = U_{э.02} = U_{э.03} = 0,75$ В, $U_{э.05} = U_{э.06} = 0,8$ В, напряжение U_2 на эмиттере $VT1$ (точка 2 или узел 2) равно $U_2 = U_{вх1} - U_{э.01} = -0,9 - 0,75 = -1,65$ В. Транзистор $VT2$ закрыт, ток I'_3 протекает по цепи: «земля», $R3, VT1, R4, U_{п.п.}$, создавая на $R3$ (точка 1) падение напряжения

$$U_1 = -I'_3 R3 = -0,9 \text{ В.}$$

Напряжение на выходе y_2

$$U_{\text{вых}2} = U_1 - U_{\text{э.о6}} = -0,9 - 0,8 = -1,7 \text{ В.}$$

Напряжение U_3 узла 3 приблизительно равно нулю. Следовательно, на выходе y_1

$$U_{\text{вых}1} = U_3 - U_{\text{э.о5}} = 0 - 0,8 = -0,8 \text{ В.}$$

Если $U_{\text{вх}1} = U_{\text{вх}2} = -1,7 \text{ В}$, то $U_{63} = U_{\text{оп}} = -1,3 \text{ В} > U_{61} = U_{62} = -1,7 \text{ В}$.

Следовательно, $VT3$ будет открыт и работать в активном режиме. В точке 2

$$U_2 = U_{\text{оп}} - U_{\text{э.о3}} = -1,3 - 0,75 = -2,05 \text{ В.}$$

Теперь $U_1 = 0$, $U_2 = -0,9 \text{ В}$ и соответственно $U_{\text{вых}2} = -0,9 \text{ В} = U^1$, $U_{\text{вых}1} = -1,7 \text{ В} = U^0$. Таким образом, ток $I_3 = 4 \text{ мА}$ переключается либо в цепь $R3$, либо в цепь $R5$ (откуда название *токовый переключатель*). Объединение коллекторов $VT1$, $VT2$ в точке 1 дает булеву функцию $x_1 x_2 = x_1 + x_2$ (см. рис. 4.11). Напряжения в точках 1 и 3 находятся в противофазе, поэтому в точке 3 реализуется функция $x_1 + x_2$. В схемах серии 500 резисторы $R9$, $R10$ отсутствуют внутри ИС и ставятся внешним монтажом. При этом используется несколько вариантов номиналов резисторов $R9$, $R10$ и источника смещения $U_{\text{см}}$: $R9 = R10 = 50 \text{ Ом}$ ($U_{\text{см}} = -2 \text{ В}$); 75 или 100 Ом ($U_{\text{см}} = -2,4 \text{ В}$); 240—500 Ом ($U_{\text{см}} = -5,2 \text{ В}$).

Для реализации сложных логических функций в ЭСЛ используется объединение эмиттеров, объединение коллекторов и каскадное соединение ТП (2- и 3-уровневые схемы).

Инжекционные интегральные логические схемы (И²Л).

В основе И²Л схем лежат два принципа схемотехники и технологии биполярных схем: 1) совмещение электрически связанных однородных областей полупроводника в кристалле ИС, что приводит к увеличению степени интеграции; 2) отказ от традиционного способа питания цепи базы и коллектора ключевых транзисторов через резисторы.

На рис. 4.21 показана инжекционная структура с дополнительным $p-n$ -переходом. Если база (область p_2) переключающего вертикального транзистора $n_2-p_2-n_1$ -типа расположена вблизи прямосмещенного перехода p_1-n_1 , то часть инжектированных данным переходом дырок попадает в область базы p_2 . В результате

нарушения электронейтральности базы этого транзистора через переход база — эмиттер начинает протекать ток, смещающий этот переход в прямом направлении. Область p_1 , введенную для инжекции избыточных носителей, называют инжектором E . Питание инжектора осуществляется от внешнего генератора тока или от источника

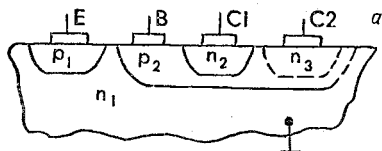


Рис. 4.21

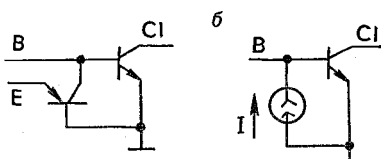


Рис. 4.22

напряжения через резистор (один внешний резистор). Данную инжекционную структуру можно представить в виде схемы, содержащей два транзистора: токозадающий горизонтальный $p_1 - n_1 - p_2$ -типа (включен по схеме с общей базой) и переключающий вертикальный $n_2 - p_2 - n_1$ -типа (рис. 4.22, а). Упрощенная эквивалентная схема состоит из транзистора $n - p - n$ -типа и генератора тока I в цепи его базы (рис. 4.22, б).

Обычно транзистор $n - p - n$ -типа выполняется с несколькими коллекторами (область n_3 на рис. 4.21).

Когда транзисторы $VT1$ и $VT2$ закрыты, транзистор $VT3$ открыт током инжектора I (сплошная линия прохождения тока на рис. 4.23) и напряжение в узле 2 ЛЭ И²Л

$$U_2 = U_{э.о} = 0,75 \text{ В} = U^1,$$

что соответствует уровню логической 1 в ПЛ. Если транзистор $VT2$ откроется, то ток инжектора $VT3$ переключится в цепь коллектора $VT2$ (пунктирная линия на рис. 4.23) и транзистор $VT3$ закроется. В узле 2 будет напряжение насыщения коллектора $VT2$:

$$U_2 = U_{к.н2} = 0,05 \text{ В} = U^0.$$

Схемы И²Л имеют $t_{зд.р} > 5 \text{ нс}$, $P_{\text{пот}} < 0,2 \text{ мВт}$, $P_{\text{пот}} \times \times t_{зд.р} < 1 \text{ пДж}$.

Площадь, приходящаяся на один ЛЭ в схемах в И²Л, приблизительно в 10 раз меньше, чем в схемах ТТЛ. Применение диодов Шотки в схемах И²Л позволяет без увеличения потребляемой мощности получить еще более

высокое быстродействие.

Элементы на полевых транзисторах. Наиболее дешевыми и одновременно самыми медленно действующими являются схемы ЛЭ на p -канальных МОП-транзисторах*. Поэтому наибольшее применение получили схемы на n -канальных МОП-транзисторах и схемы, содержащие

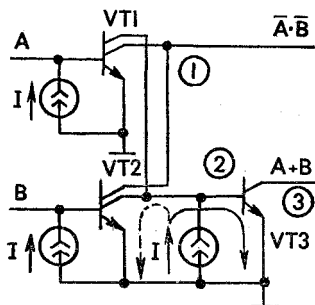


Рис. 4.23

как n -, так и p -канальные МОП-транзисторы, т. е. схемы на дополняющих транзисторах, которые называют также элементами КМОП (элементы на комплементарных** МОП-транзисторах). По быстродействию схемы ЛЭ на полевых транзисторах уступают схемам ЛЭ на биполярных транзисторах.

На рис. 4.24, а показана базовая логическая схема микропроцессорного комплекта серии К580. Транзистор $VT2$ выполняет роль нагрузочного резистора. Он постоянно открыт и работает в крутой области ВАХ.

При подаче, например, на вход x_1 напряжения $U_{вх1}^1 = 5$ В, а на вход x_2 напряжения $U_{вх2}^0 \approx 0$ транзистор $VT1$ открыт и работает в крутой области ВАХ, транзистор $VT3$ — закрыт, на выходе $U_{вых}^0 \approx 0$.

Если на оба входа x_1 и x_2 поступят напряжения $U_{вх1}^0 = U_{вх2}^0 \approx 0$, то транзисторы $VT1$ и $VT2$ будут закрыты, а $U_{вых}^1$, которое появилось на выходе через открытый $VT3$, будет равно 5 В.

На рис. 4.24, б приведен базовый элемент микропроцессорного комплекта с одним источником питания. Транзистор $VT2$ со встроенным каналом n -типа выполняет

* Иногда вместо МОП (металл-оксид-полупроводник) встречается название МДП-транзистор (металл-диэлектрик-полупроводник). Будем считать эти названия равнозначными.

** От англ. complementary — дополнительный.

роль токостабилизирующей нагрузки. Когда один из транзисторов ($VT1$ или $VT3$) открыт, то транзистор $VT2$ работает в пологой области ВАХ. При закрытых $VT1$ и $VT3$ транзистор $VT2$ работает в крутой области ВАХ, обеспечивая $U_{\text{вых}}^1 = 5 \text{ В}$.

Основным достоинством ИС на КМОП-транзисторах является микроваттная потребляемая мощность в статическом режиме, высокая помехоустойчивость, высокая нагрузочная способность, а также способность работать в широком диапазоне питающих напряжений.

На рис. 4.25, а показан инвертор на КМОП-транзисторах, где $VT1$ — транзистор с индуцированным каналом p -типа, а $VT2$ — n -типа. $U^1 \approx +E$, $U^0 \approx 0$. Когда на вход x подается $U_{\text{вх}}^1 = U_3^{(n)} = +E$, то транзистор $VT2$ открывается и работает в крутой области ВАХ, а $VT1$ — закрыт, так как его напряжение затрат — исток $U_3^{(p)}$ равно нулю (на затворе $U_{\text{вх}}^1 = +E$ и на истоке $+E$, следовательно, разность равна нулю). На выходе y в этом случае $U_{\text{вых}}^0 = 0$. При подаче $U_{\text{вх}}^0 = 0$ транзистор

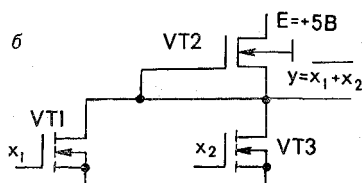
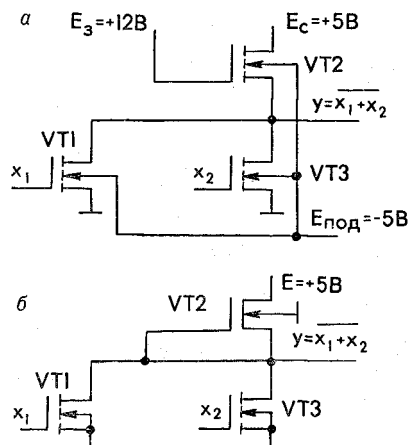


Рис. 4.24

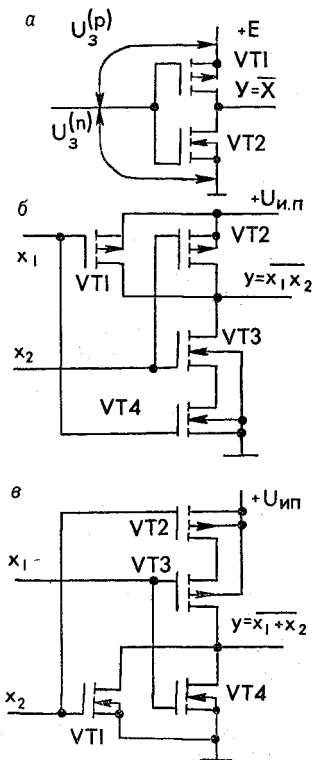


Рис. 4.25

$VT2$ запирается, так как его $U_3^{(n)} = 0$, а транзистор $VT1$ будет открыт и работать в крутой области ВАХ, так как $U_3^{(p)} = 0 - E = -E$. На выходе y $U_{\text{вых}}^1 = E$. Таким образом, как при $U_{\text{вх}}^1$, так и при $U_{\text{вх}}^0$ один из транзисторов всегда закрыт, в статическом состоянии (когда на выходе 0 или 1) в схеме не протекает сквозной ток от $+E$ к «земле».

На рис. 4.25, б, в приведены схемы И — НЕ, ИЛИ — НЕ на КМОП-транзисторах. На схеме рис. 4.25, б при подаче, например, $U_{\text{вх}1}^1 = +U_{\text{н.п.}}$, $U_{\text{вх}2}^0 = 0$ транзисторы $VT4$ и $VT2$ будут открыты, а $VT1$ и $VT3$ — закрыты и на выходе $U_{\text{вых}}^1 = +U_{\text{н.п.}}$ (через открытый $VT2$).

Следует отметить, что с увеличением частоты входных сигналов элементы КМОП начинают потреблять динамическую мощность на перезарядку емкости нагрузки. Гарантированная помехоустойчивость $U_{\text{п}}^{\pm}$ для элементов КМОП серии 564 (561) составляет 30 % от $U_{\text{н.п.}}$, что значительно выше, чем в схемах ТТЛ, ЭСЛ. Свободные входы КМОП-схем следует подключать либо к «земле» (в схемах ИЛИ, ИЛИ — НЕ), либо к источнику питания (в схемах И, И — НЕ).

4.2. Триггеры

Общие положения. Двоичный статический триггер имеет два устойчивых состояния 0 и 1 и два выхода: прямой Q и инверсный \bar{Q} . Когда $Q = 0$, $\bar{Q} = 1$, триггер находится в нулевом состоянии, при $Q = 1$, $\bar{Q} = 0$ триггер — в единичном состоянии. Если в некоторый момент времени t_n на триггер, находящийся в состоянии Q_n , поступают информационные сигналы, то это приводит к переходу триггера в следующий момент времени t_{n+1} в состояние Q_{n+1} , т. е.

$$Q_{n+1} = f(x_n, Q_n).$$

Зависимость Q_{n+1} не только от x_n , но и от Q_n , а также наличие в некоторых типах триггеров запрещенных комбинаций по x_n приводит к тому, что возможны пять логических состояний выходного сигнала Q_{n+1} :

- 0 — триггер находится в нулевом состоянии;
- 1 — триггер находится в единичном состоянии;
- Q_n — состояние триггера не изменяется при изменении входных сигналов x_i n ;

\bar{Q}_n — состояние триггера изменяется на противоположное при изменении входных сигналов;

\emptyset — неопределенное состояние (иногда обозначают Н/0, *, X). Обозначение \emptyset похоже на ноль с наложенной на него единицей, т. е. неизвестно, триггер установится в состояние 1 или в 0.

Триггер имеет следующие входы и выходы (рис. 4.26):
 S_y, R_y — установочные входы;
 $x_1 — x_m$ — информационные сигналы, определяющие, в какое состояние необходимо установить триггер, чтобы

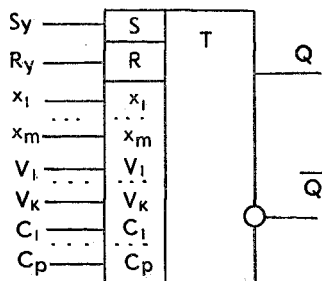


Рис. 4.26

на выходах (прямом Q , инверсном \bar{Q}) было одно из пяти перечисленных выше состояний выходного сигнала Q_{n+1} ;

$C_1 — C_p$ — синхронизирующие входы; они определяют момент времени t_{n+1} , в который осуществляется переход триггера в то состояние, которое задано информационными входами $x_1 — x_m$;

$V_1 — V_k$ — управляющие входы, разрешающие прохождение либо информационных сигналов $x_1 — x_m$, либо синхросигналов по входам $C_1 — C_p$ на триггер.

В зависимости от используемых информационных входов x_i триггеры подразделяются на следующие типы:

RS-триггеры, которые имеют вход установки в состояние 1 S (от англ. set — установка) и вход установки в 0 R (от англ. reset — сброс, установка в 0). Триггеры, имеющие инверсные входы \bar{R} и \bar{S} , называют *RS-триггерами с инверсными входами*;

JK-триггеры, имеющие информационный вход установки в состояние 0 K (от англ. kill — внезапное отключение) и вход установки в 1 J (от англ. jess — внезапное включение);

D-триггеры (от англ. delay — задержка), имеющие один информационный вход D , по которому они устанавливаются как в состояние 1, так и в 0;

DV-триггеры, работающие как *D*-триггеры при $V = 1$ и сохраняющие предыдущее состояние Q_n , если $V = 0$ (управляющий вход V от англ. valve — клапан);

T-триггеры — счетные триггеры, имеющие информационный вход T (от англ. toggle — релаксатор).

В табл. 4.2 приведена сокращенная таблица состояний триггеров.

Таблица 4.2

Входные сигналы в момент времени t_n		Состояние схемы Q_{n+1} в момент времени t_{n+1}					
$S, \bar{S},$ J, D	R, \bar{R}, K, T, V	$\bar{R}\bar{S}$	RS	JK	D	DV	T
0	0	\emptyset	Q_n	Q_n	0	Q_n	Q_n
0	1	1	0	0	0	0	\bar{Q}_n
1	0	0	1	1	1	Q_n	Q_n
1	1	Q_n	\bar{Q}_n	\bar{Q}_n	1	1	\bar{Q}_n

✓ По наличию синхровходов триггеры подразделяются на асинхронные и синхронные.

Если триггер не имеет ни одного синхровхода C (от англ. clock — первичный источник сигналов синхронизации), то он называется *асинхронным* и момент изменения выходного сигнала определяется моментом прихода сигналов x_i . ✓

✓ *Синхронные* триггеры принято классифицировать по способу приема входной информации и по принципу передачи принятой информации на выход. По способу приема информации можно выделить две группы: управляемые уровнем синхросигнала (*триггеры со статическим управлением*) и управляемые фронтом синхросигнала (*триггеры с динамическим синхронизирующим входом*). ✓

✓ Синхронный триггер, управляемый уровнем синхросигнала, принимает те информационные сигналы, которые появляются на его входах в течение всей длительности импульса синхронизации. Синхронный триггер, управляемый фронтом синхросигнала, принимает те информационные сигналы, которые совпадают с приходом только фронта синхросигнала. ✓

✓ По принципу передачи принятой информации синхронные триггеры подразделяют на триггеры с одной (*одноступенчатые*) и двумя (*двухступенчатые*) ступенями запоминания информации. У синхронных одно-

ступенчатых триггеров прием и передача на выход принятой информации неразрывно связаны. У синхронных двухступенчатых триггеров прием и передача на выход принятой информации разделены так, что потенциалы на выходах триггера меняются лишь после того, как триггер перейдет в режим хранения принятой информации. Двухступенчатые триггеры строят с использо-

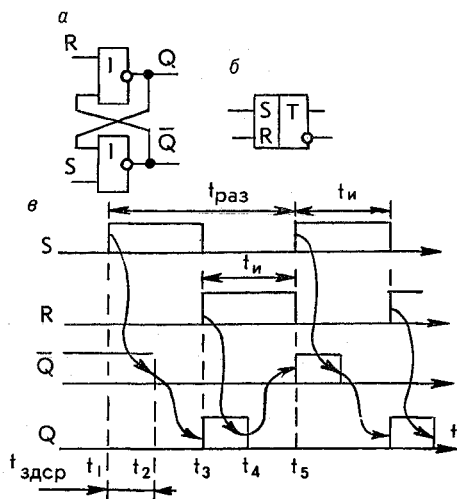


Рис. 4.27

ванием основного m (от англ. master — основной) и вспомогательного s (от англ. slave — вспомогательный) триггеров.

Правила (законы) функционирования триггеров могут быть заданы: 1) словесным описанием; 2) таблицей переходов триггера, т. е. таблицей информационных значений входных сигналов x_i , внутренних состояний Q_n и выходных сигналов Q_{n+1} триггера; 3) характеристическими уравнениями — логическими функциями типа $Q_{n+1} = f(Q_n, x_{im})$, где $i = 1, 2, \dots, m$; 4) в виде графа.

RS-триггеры. Закон функционирования RS-триггера отражен в табл. 4.3. На рис. 4.27, *а* показана логическая структура (функциональная схема) RS-триггера, на рис. 4.27, *б* — его условное графическое обозначение (УГО), а на рис. 4.27, *в* — график предельного динамического режима работы RS-триггера. Примем для простоты $t_{зд.р}^{1,0} \approx t_{зд.р}^{0,1} \approx t_{зд.р.ср}$.

В момент t_1 на вход триггера поступают сигналы $S=1, R=0$. Поступление сигнала $S=1$ приводит к переключению с задержкой $t_{зд.р}^{1,0} \approx t_{зд.р.ср}$ в момент t_2 нижнего элемента ИЛИ — НЕ (табл. 4.4) и на выходе \bar{Q} устанавливается 0. Возникновение в момент t_2 сигнала $\bar{Q}=0$ при $R=0$ приводит к появлению с задержкой

Таблица 4.3

R	S	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	\emptyset

Таблица 4.4

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	0

$Q=1$ в момент t_3 . Быстродействие триггера определяет наименьший разрешающий интервал времени $t_{раз}$ между входными сигналами минимальной длительности, действующими на тот же вход триггера и вызывающими бесперебойное переключение триггера.

Триггер переключается «бесперебойно», если длительность выходного сигнала не менее $t_{зд.р.ср}$ одного ЛЭ схемы. Из рис. 4.27, в

$$t_{раз} = 4t_{зд.р.ср}.$$

Максимальная частота переключения триггера

$$f_{max} = 1/t_{раз}.$$

Так как мы не учитывали длительность фронтов, то фактическая длительность выходного сигнала Q (\bar{Q}) меньше $t_{зд.р.ср}$, поэтому рабочей принято считать частоту в 1,5 раза меньшую:

$$f_{раб} = f_{max}/1,5.$$

Минимальная длительность входного сигнала

$$t_{и} = \sum_{i=1}^k t_{зд.р.ср i}$$

где k — число элементов в цепочке от входа информационного (или тактового) сигнала до входа элемента, на котором замыкается триггерное кольцо обратной

связи. В нашем случае $t_{\text{и}} = 2t_{\text{зд.р.ср}}$, т. е. задержке срабатывания двух элементов ИЛИ — НЕ. В дальнейшем на временных диаграммах не будем учитывать наличие задержки ЛЭ, т. е. будем считать $t_{\text{зд.р.ср}} = 0$ (см. рис. 4.28).

На графике (рис. 4.28) приведен пример произвольного изменения входных сигналов S и R . В момент t_3

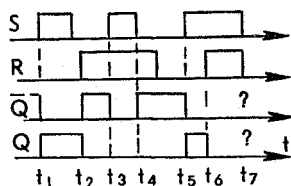


Рис. 4.28

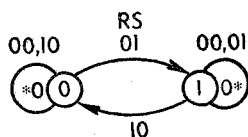


Рис. 4.29

на входы поступают сигналы $S = R = 1$, на выходах $Q = \bar{Q} = 0$. $Q = \bar{Q} = 0$ — это вполне определенное состояние, появление которого мы можем предсказать.

Неопределенность \emptyset (см. табл. 4.3) функционирования RS-триггера наступит в момент t_7 (рис. 4.28), когда оба сигнала $R = S = 1$ одновременно изменятся на $R = S = 0$. В этом случае из-за неидентичности элементов ИЛИ — НЕ триггер перейдет в состояние $Q = 1$, $\bar{Q} = 0$ или $Q = 0$, $\bar{Q} = 1$, однако в какое именно, сказать заранее невозможно.

На рис. 4.29 закон функционирования RS-триггера представлен в виде графа: в кружках указаны состояния триггера, на дугах графа (направленных ребрах) проставлены комбинации входных сигналов R и S , * — безразличное состояние (0 или 1).

$\bar{R}\bar{S}$ -триггер. На рис. 4.30, а показана функциональная схема RS-триггера с инверсными входами, на рис. 4.30, б — его УГО. Состояния триггера в зависимости от комбинаций входных сигналов приведены в табл. 4.5. Произвольная комбинация входных сигналов \bar{S} и \bar{R} и изменения выходов Q и \bar{Q} показаны на рис. 4.31.

На рис. 4.32, а дана функциональная схема RS-триггера с инверсными входами, имеющая по два фактических входа \bar{R}_1, \bar{R}_2 и \bar{S}_1, \bar{S}_2 , а на рис. 4.32, б — его УГО.

По количеству информационных входов триггеры подразделяются на *одновходовые, двухходовые и много-*

входовые. Наибольшее распространение получили одно-входовые и двухвходовые триггеры. Не следует путать количество информационных входов с количеством фактических входов, на которые поступают информацион-

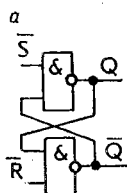


Рис. 4.30

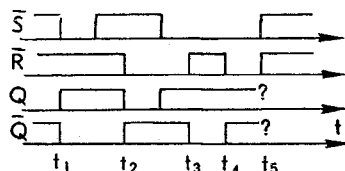
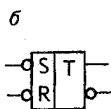


Рис. 4.31

ные сигналы. Для рис. 4.32 $\bar{S} = \bar{S}_1 \bar{S}_2$, $\bar{R} = \bar{R}_1 \bar{R}_2$, т. е. триггер имеет два информационных \bar{S} , \bar{R} и 4 фактических входа (\bar{S}_1 , \bar{S}_2 , \bar{R}_1 , \bar{R}_2), по которым поступают инфор-

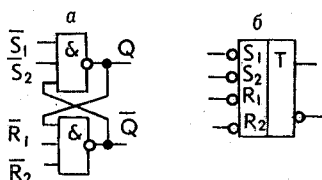
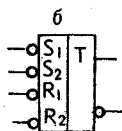


Рис. 4.32



мационные сигналы, и функционирует в соответствии с таблицей состояний (табл. 4.5).

Таблица 4.5

t_n		t_{n+1}
\bar{S}	\bar{R}	Q_{n+1}
0	0	\emptyset
0	1	1
1	0	0
1	1	Q_n

На рис. 4.33 показан граф $\bar{R}\bar{S}$ -триггера.

Синхронные триггеры, кроме информационных входов, имеют также синхровход C . На рис. 4.34, а приведена функциональная схема синхронного RS-триггера, на рис.

4.34, б — его УГО, а на рис. 4.34, в — пример временной диаграммы работы.

На рис. 4.35, а показана функциональная схема синхронного RS-триггера с логикой 4И на входе и асинхронными входами \bar{R} , \bar{S} , а на рис. 4.35, б — его УГО. Если

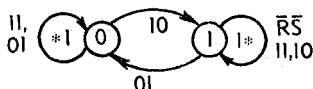


Рис. 4.33

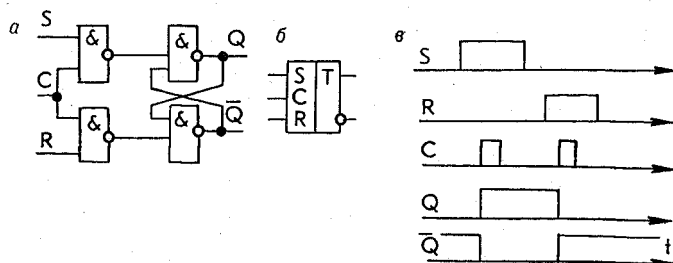


Рис. 4.34

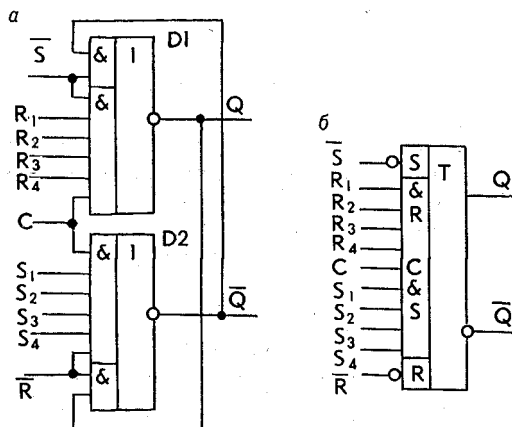


Рис. 4.35

$\bar{S} = \bar{R} = 1$, то триггер работает как синхронный RS-триггер, где $R = R_1 R_2 R_3 R_4$, а $S = S_1 S_2 S_3 S_4$ в соответствии с табл. 4.3.

Для всех наборов $\bar{R}\bar{S}$, кроме $\bar{S} = \bar{R} = 1$, триггер работает как асинхронный $\bar{R}\bar{S}$ -триггер в соответствии с

табл. 4.5 независимо от состояния входов $C, R_1, \dots, R_4, S_1, \dots, S_4$.

D-триггер. Закон функционирования D-триггера, приведенный в табл. 4.6, описывается логическим уравнением

$$Q_{n+1} = D_n.$$

Асинхронный D-триггер не нашел применения, так как представляет собой последовательное соединение двух инверторов.

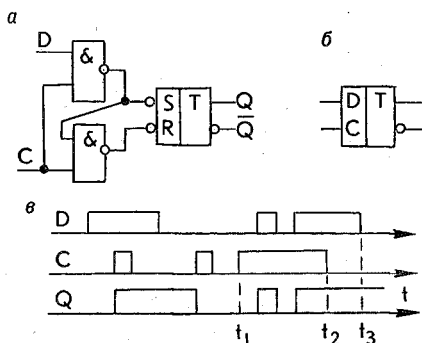


Рис. 4.36

На рис. 4.36, *а* приведена логическая структура синхронного D-триггера, на рис. 4.36, *б* — его УГО. Это одноступенчатый триггер, управляемый уровнем синхросигнала: на всей длительности уровня синхросигнала от t_1 до t_2 выход меняется в соответствии с изменением сигнала по входу D . При $C = 0$ синхронный триггер не реагирует на изменение сигнала по входу D . Пример временной диаграммы дан на рис. 4.36, *в*.

Таблица 4.6

t_n	t_{n+1}
D	Q_{n+1}
0	0
1	1

DV-триггер. На рис. 4.37, *а* приведена функциональная схема синхронного одноступенчатого DV-триггера,

а на рис. 4.37, б — его УГО. При $V=1$ DV-триггер работает как синхронный D -триггер. Когда на управляющем входе $V=0$, триггер сохраняет предыдущее состояние Q_n .

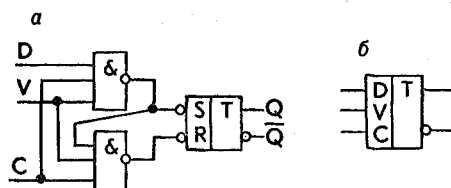


Рис. 4.37

D-триггер с динамическим синхронизирующим входом.

На рис. 4.38 показана функциональная схема D -триггера с динамическим синхронизирующим входом C (триггер, управляемый фронтом синхросигнала C). Этот D -триггер состоит из трех $\overline{R}\overline{S}$ -триггеров: двух входных триггеров $T1$ (вентили $G1, G2$), $T2$ (вентили $G3, G4$) и одного выходного $T3$ (вентили $G5, G6$).

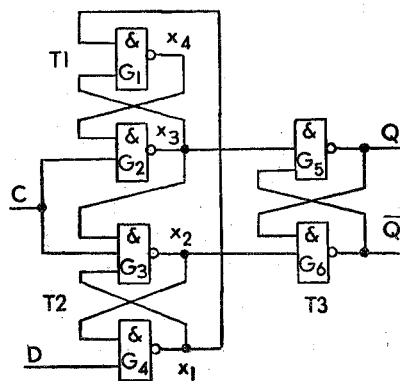


Рис. 4.38

При $C=0$ сигналы x_2, x_3 принимают единичное значение ($x_2=x_3=1$) и выходной триггер $T3$ работает в режиме хранения информации, а триггеры $T1, T2$ — в режиме приема информации по входу D . При этом

$$x_1 = x_2 \cdot D = 1 \cdot D = D, \quad x_4 = x_1 x_3 = D \cdot 1 = D.$$

Это означает, что всегда в одном из двух триггеров ($T1$ либо $T2$) оба выходных сигнала равны 1. Неправильное состояние исчезает при поступлении положительного фронта синхросигнала C (при переходе C с 0 на 1). Во время положительного фронта тактирующего импульса выполняется равенство $Q = D$. Правильное состояние приводит к запрету прохождения сигнала D на входы

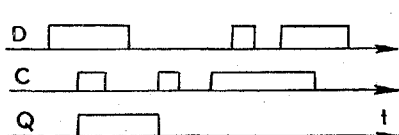


Рис. 4.39

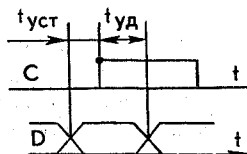


Рис. 4.40

триггера $T3$. Для записи новой (например, единичной) информации необходимо подать $C = 0, D = 1$. С приходом положительного фронта $C = 0 \rightarrow 1$ триггер $T3$ переключается в единичное состояние ($Q = 1, \bar{Q} = 0$).

На рис. 4.39 показаны для сравнения входные сигналы такой же формы, как и на рис. 4.36, в. Из сравнения выходов Q видна разница в переключении триггера, управляемого уровнем (см. рис. 4.36, а), и триггера, управляемого фронтом (см. рис. 4.38) синхросигнала. Для устойчивой работы D-триггера с динамическим входом в момент действия положительного фронта синхроимпульса сигнал по входу D не должен изменяться (объединять входы C и D не допускается). Время установления сигнала $t_{уст}$ (для микросхемы К155ТМ2) на входе D должно быть не менее 20 нс, а время удержания $t_{уд}$ не менее 5 нс (рис. 4.40). На рис. 4.40 показано, что в течение времени $t_{уст} + t_{уд}$ сигнал по входу D не должен меняться.

JK-триггер. В качестве примера на рис. 4.41 приведена упрощенная схема синхронного двухступенчатого JK-триггера серии К155ТВ1 с логикой 3И на входе и асинхронными входами \bar{R}, \bar{S} .

JK-триггер состоит из основного триггера m (см. рис. 4.35, б) с выходами Q_1, \bar{Q}_1 и вспомогательного (см. рис. 4.32, б) с выходами Q, \bar{Q} . В табл. 4.7 приведен закон функционирования этого триггера, где $J = J_1 J_2 J_3$, $K = K_1 K_2 K_3$.

На рис. 4.42, а приведен пример временной диаграмм-

мы работы JK-триггера микросхемы К155ТВ1. По установочным входам \bar{R} , \bar{S} JK-триггер работает как асинхронный RS-триггер с инверсными входами ($\bar{R}\bar{S}$ -триггер) в соответствии с табл. 4.5 независимо от значений сигналов по входам C , J_1 , ..., J_3 , K_1 , ..., K_3 . В табл. 4.7 крестиком отмечены безразличные значения (0 или 1) по входам J и K .

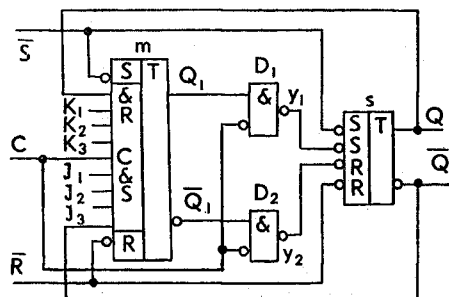


Рис. 4.41

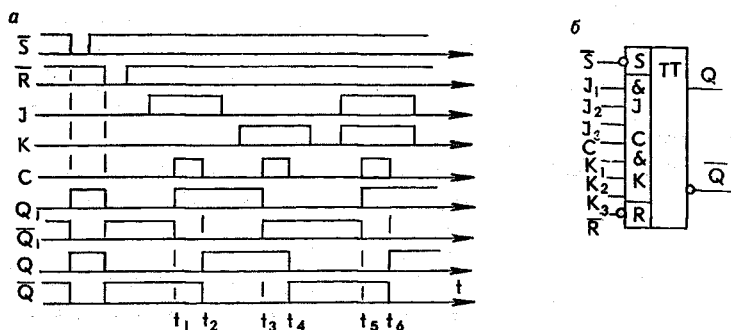


Рис. 4.42

При $\bar{R} = \bar{S} = 1$ триггер работает как синхронный двухступенчатый JK-триггер ms , т. е. по переднему (положительному) фронту синхроимпульса, например в момент t_1 триггер m устанавливается в 1 ($Q_1 = 1$, $\bar{Q}_1 = 0$), так как $J = 1$, $K = 0$), а по заднему (отрицательному) фронту синхроимпульса по входу C информация с триггера m переписывается с помощью элементов $D1$, $D2$ в триггер s и появляется на выходах $Q = 1$, $\bar{Q} = 0$.

Таблица 4.7

t_n				t_{n+1}
\bar{R}	\bar{S}	J	K	Q_{n+1}
0	0	×	×	0
0	1	×	×	0
1	0	×	×	1
1	1	0	0	Q_n
1	1	0	1	0
1	1	1	0	1
1	1	1	1	\bar{Q}_n

Начиная с момента t_2 (до t_3) $C=0$,

$$y_1 = \overline{Q_1 \bar{C}} = \overline{Q_1 0} = \overline{Q_1 1} = \bar{Q}_1;$$

$$y_2 = \overline{\bar{Q}_1 \bar{C}} = \overline{\bar{Q}_1 0} = \overline{\bar{Q}_1 1} = Q_1.$$

Так как триггер s — это RS-триггер с инверсными входами, то выполняется условие $Q = Q_1$, $\bar{Q} = \bar{Q}_1$.

При $\bar{R} = \bar{S} = 1$ и $C = 1$ $y_1 = y_2 = 1$, триггер s сохраняет предыдущее состояние. При $J = K = 1$ благодаря обратным связям с выходов Q , \bar{Q} на вход триггера m осуществляется инверсия предыдущего состояния \bar{Q}_n .

На рис. 4.42, б дано УГО JK-триггера микросхемы К155ТВ1. Две буквы ТТ в УГО показывают, что триггер двухступенчатый.

Т-триггер. Это счетный триггер, который меняет свое состояние на обратное при $T = 1$ (см. табл. 4.2).

На рис. 4.43, а приведена схема асинхронного Т-триггера на базе JK-триггера, на рис. 4.43, б, в — возможные УГО Т-триггера, а на рис. 4.43, д — временная диаграмма работы. Так как JK-триггер двухступенчатый, то переключение на выходе Q осуществляется по заднему отрицательному фронту импульса по входу T , который подается на вход C JK-триггера. При построении Т-триггера используется в данном случае переход JK-триггера при $J = K = 1$ в состояние Q_n (см. табл. 4.2 или 4.7). Асинхронный Т-триггер на базе JK-триггера можно получить также, подав на входы $J = K = 1$ уровни логической 1 (рис. 4.43, г).

На рис. 4.44, а, б, в показаны соответственно реализация синхронного Т-триггера на базе JK-триггера, его

УГО и временная диаграмма работы. На рис. 4.45, а, б даны реализация асинхронного Т-триггера на базе D-триггера с динамическим синхронизирующим входом и временная диаграмма работы. Так как D-триггер с динамическим входом (см. рис. 4.38) срабатывает по переднему фронту синхроимпульса, то и изменения на выходах

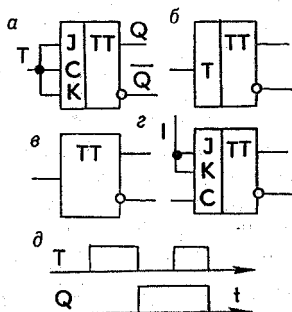


Рис. 4.43

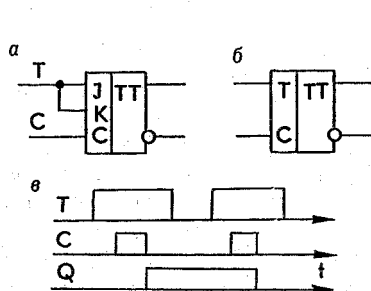


Рис. 4.44

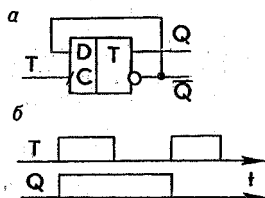


Рис. 4.45

Q и \bar{Q} в схеме, показанной на рис. 4.45, а, будут по переднему (положительному) фронту импульса по входу T .

4.3. Операционные узлы

Общие сведения. Операционные (функциональные) узлы предназначены для выполнения простейших операций над машинными словами или их частями: запоминание, сдвиг кода, сложение кодов, их сравнение и т. д. Операционные узлы строятся из логических и (или) запоминающих элементов (триггеров). К типовым операционным узлам ЦВМ относятся дешифраторы, шифраторы, мультиплексоры, демультиплексоры, схемы сравнения (компараторы), схемы свертки, сумматоры, регистры, счетчики и др.

Дешифратор. Полный двоичный дешифратор, или декодер (от англ. decoder), — это операционный узел ЦВМ комбинационного типа, преобразующий n -разрядный двоичный позиционный код $X = x_{n-1}, \dots, x_2x_1x_0$ в m -разрядный унитарный код, где $m = 2^n$. На любом наборе входных переменных единица появляется только на одном из выходов, при нулях на остальных $m - 1$ выходах (или наоборот, если дешифратор имеет не прямые, а инверсные выходы).

Дешифраторы применяются в устройствах управления для дешифрации операций или микрокоманд в управляющие сигналы, в запоминающих устройствах для выбора ячейки памяти при записи или считывании информации и др. В соответствии с таблицей истинности двухразрядного ($n = 2$) дешифратора (табл. 4.8),

Таблица 4.8

x_1	x_0	y_3	y_2	y_1	y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

имеющего два входа x_1 и x_0 и четыре ($m = 2^n = 2^2 = 4$) выхода y_3, y_2, y_1, y_0 , можем записать

$$y_0 = \bar{x}_1\bar{x}_0, y_1 = \bar{x}_1x_0, y_2 = x_1\bar{x}_0, y_3 = x_1x_0. \quad (4.3 \text{ а})$$

Индекс i выходных сигналов y_i является десятичным эквивалентом двоичного кода X . Например, $i = 2$ для y_2 определяется как

$$i = x_12^1 + \bar{x}_02^0 = 1 \cdot 2^1 + 0 \cdot 2^0 = 2.$$

По наличию входа C дешифраторы подразделяются на нестробируемые и стробируемые.

В соответствии с системой булевых функций (4.3) на рис. 4.46, а, б, в показаны соответственно функциональная схема дешифратора, УГО нестробируемого и стробируемого дешифраторов. Инверторы $D1, D2$ и $D3, D4$ называют адресными инверторами или адресными формирователями. Они предназначены для того, чтобы каждый вход (x_0, x_1) представлял собой одну единичную нагрузку. Если добавить дополнительный стро-

бирующий вход C (пунктир на рис. 4.46, а), то получим схему стробируемого дешифратора, который при $C=1$ работает в соответствии с таблицей истинности 4.8, а при $C=0$ $y_0=y_1=y_2=y_3=0$.

Для стробируемого дешифратора

$$y_0 = \bar{x}_1 \bar{x}_0 C, \quad y_1 = \bar{x}_1 x_0 C, \quad y_2 = x_1 \bar{x}_0 C, \quad y_3 = x_1 x_0 C. \quad (4.36)$$

По входу C также ставят инвертор, аналогичный $D1-D4$ (здесь для простоты схемы он не показан). Пря-

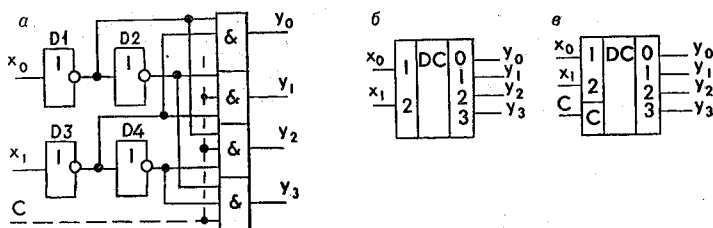


Рис. 4.46

мая реализация системы (4.3) приводит к структуре, показанной на рис. 4.46, а и называемой *одноступенчатым (линейным) дешифратором*.

Для построения многоразрядных дешифраторов применяют каскадное соединение дешифраторов меньшей разрядности. На рис. 4.47 приведена схема 4-разрядного стробируемого дешифратора на базе 2-разрядных стробируемых дешифраторов (рис. 4.46, в).

Демультимплексор. Демультимплексор (от англ. demultiplexer) — операционный узел ЦВМ, осуществляющий микрооперацию передачи сигнала с одного информационного входа на один из выходов. Демультимплексор имеет один информационный вход x , k адресных входов a_{k-1}, \dots, a_n , $n=2^k$ выходов. Для демультимплексора при $k=2$, $n=4$

$$y_0 = \bar{a}_1 \bar{a}_0 x, \quad y_1 = \bar{a}_1 a_0 x, \quad y_2 = a_1 \bar{a}_0 x, \quad y_3 = a_1 a_0 x. \quad (4.4)$$

Из сравнения выражений (4.4) и (4.3) видно, что функцию демультимплексора реализует стробируемый дешифратор, если на вход C подать x , а входы дешифратора x_1, x_0 использовать как адресные входы демультимплексора.

С помощью демультимплексора возможно распределение одного входного сигнала по нескольким различ-

ным адресам и преобразование информации из последовательной формы в параллельную.

Шифратор. Шифратор, или кодер (от англ. coder), — операционный узел ЦВМ, выполняющий функцию, обратную дешифратору, т. е. преобразует унитарный код в двоичный позиционный. Он имеет $m - 1$ входов и n выходов ($m = 2^n$). При подаче сигнала на один из

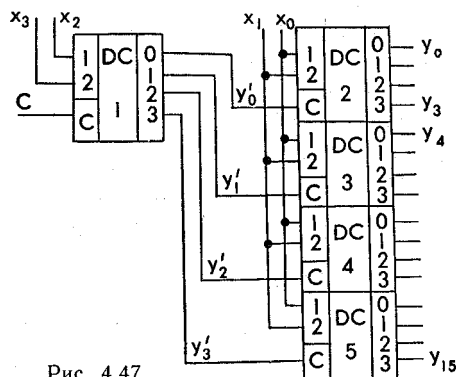


Рис. 4.47

входов (только на один) на выходе появляется двоичный код номера возбужденного входа. Закон функционирования шифратора для $n = 2$ приведен в табл. 4.9.

Таблица 4.9

x_3	x_2	x_1	x_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

По табл. 4.9 можем записать

$$y_0 = x_1 + x_3, \quad y_1 = x_2 + x_3. \quad (4.5)$$

Функциональная схема шифратора приведена на рис. 4.48, а, его УГО — на рис. 4.48, б.

Мультиплексор. Мультиплексор (от англ. multiplexer) — операционный узел ЦВМ, осуществляющий микрооперацию передачи сигнала с любого информационного входа на один выход. Мультиплексор осуществляет функцию, обратную функции демультиплексора. При по-

мощи k адресных входов a_{k-1}, \dots, a_0 в мультиплексоре можно выбирать один из $n = 2^k$ информационных сигналов $x_{n-1}, \dots, x_i, \dots, x_0$ для передачи его на один выход. С выходом соединяется тот вход, индекс i которого равен десятичному значению двоичного числа, определяемого

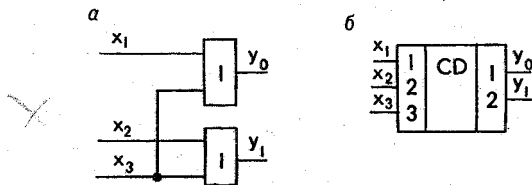


Рис. 4.48

адресными переменными. Например, мультиплексор на 4 входа имеет переключательную функцию

$$y = a_1 a_0 x_3 + a_1 \bar{a}_0 x_2 + \bar{a}_1 a_0 x_1 + \bar{a}_1 \bar{a}_0 x_0. \quad (4.6)$$

При $a_1 = 1, a_0 = 0$

$$y = 1 \cdot 0 \cdot x_3 + 1 \cdot \bar{0} \cdot x_2 + \bar{1} \cdot 0 \cdot x_1 + \bar{1} \cdot \bar{0} \cdot x_0 = x_2;$$

$$i = 2^1 \cdot a_1 + 2^0 \cdot a_0 = 2^1 \cdot 1 + 2^0 \cdot 0 = 2.$$

Мультиплексор применяют: для преобразования параллельных цифровых кодов в последовательные с целью экономии числа контактов и линий связи; на выходах блоков при считывании информации по одной разрядной шине; в многоразрядных сдвигателях информации; для реализации различных булевых функций.

В соответствии с булевой функцией (4.6) на рис. 4.49, а, приведена функциональная схема мультиплексора на 4 канала со стробированием, а на рис. 4.49, б — его УГО*.

При $\bar{S} = 0$ мультиплексор функционирует в соответствии с выражением (4.6), при $\bar{S} = 1$ $y = 0$ независимо от входов x_i и a_j .

Для мультиплексора со стробированием (рис. 4.49, а)

$$y = \bar{S} a_1 a_0 x_3 + \bar{S} a_1 a_0 x_2 + \bar{S} \bar{a}_1 a_0 x_1 + \bar{S} \bar{a}_1 \bar{a}_0 x_0 =$$

$$= \bar{S} (a_1 a_0 x_3 + a_1 \bar{a}_0 x_2 + \bar{a}_1 a_0 x_1 + \bar{a}_1 \bar{a}_0 x_0).$$

Выражение (4.6) можно записать следующим образом:

$$y = y'_3 x_3 + y'_2 x_2 + y'_1 x_1 + y'_0 x_0, \quad (4.7)$$

* Иногда говорят мультиплексор «четыре в один» (4—1).

где $y'_3 = a_1 a_0$, $y'_2 = a_1 \bar{a}_0$, $y'_1 = \bar{a}_1 a_0$, $y'_0 = \bar{a}_1 \bar{a}_0$ — выходы двухразрядного двоичного дешифратора. С учетом (4.7) функциональная схема мультиплексора приведена на рис. 4.50, а, его УГО — на рис. 4.50, б.

На рис. 4.51 показан пример построения комбинационного устройства сдвига на четырех мультипликаторах MUX 4—1, а в табл. 4.10 — закон его функционирования. При изменении адреса, например с $a_1 = 0, a_0 = 0$ на $\bar{a}_1 = 0, a_0 = 1$ на выходах $F_3 — F_0$ появляется сдвинутый на один разряд влево код X .

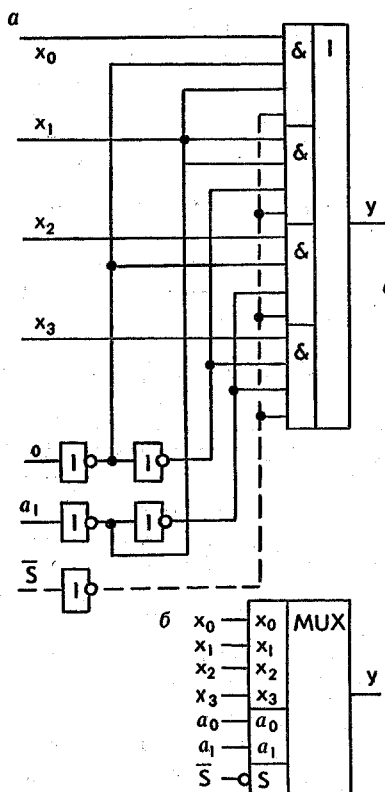


Рис. 4.49

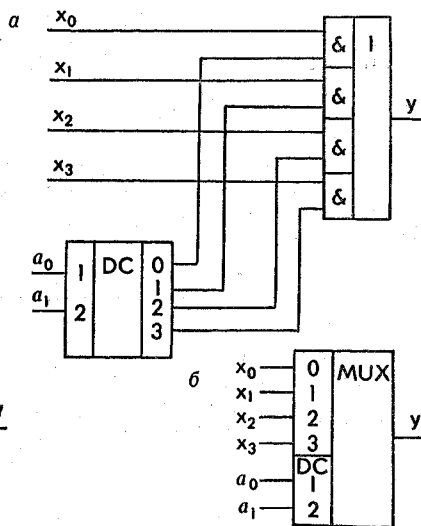


Рис. 4.50

С помощью мультиплексора можно реализовать любые булевы функции адресных переменных. Закон функционирования MUX 4—1 (рис. 4.50) отражен в табл. 4.11. Там же приведены значения функции y_1 сложения по модулю 2 адресных переменных. Для реализации функции y_1 необходимо на входы мультиплексора подать сигналы $x_0 = x_3 = 0, x_1 = x_2 = 1$ (рис. 4.52).

Таблица 4.10

a_1	a_0	F_3	F_2	F_1	F_0
0	0	x_6	x_5	x_4	x_3
0	1	x_5	x_4	x_3	x_2
1	0	x_4	x_3	x_2	x_1
1	1	x_3	x_2	x_1	x_0

Таблица 4.11

a_1	a_0	y	$y_1 = a_1 \oplus a_0$
0	0	x_0	0
0	1	x_1	1
1	0	x_2	1
1	1	x_3	0

При необходимости увеличения числа входов x_i мультиплексора можно применить, например, каскадное соединение. На рис. 4.53 приведен пример построения MUX 16—1 из MUX 4—1.

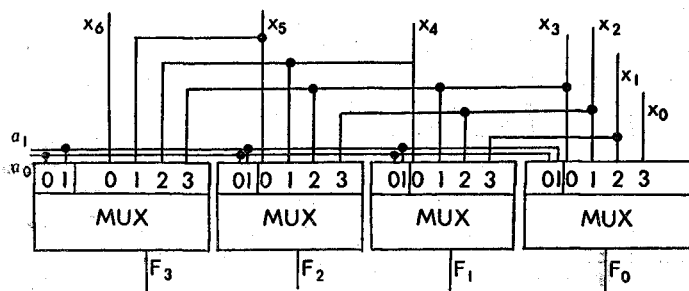


Рис. 4.51

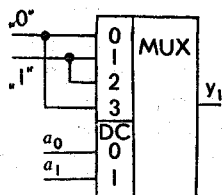


Рис. 4.52

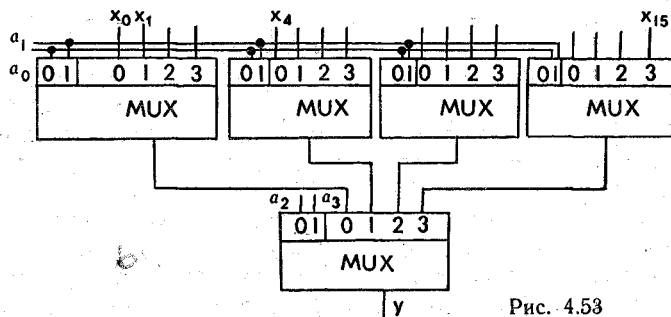


Рис. 4.53

Схема сравнения. Схема сравнения, или компаратор (от англ. compare — сравнивать), — операционный узел ЦВМ, предназначенный для сравнения двух чисел A и B . Результатом сравнения является обнаружение состояний $A = B$, $A > B$ или $A < B$.

Таблица 4.12

a	b	$y_{a>b}$	$y_{a=b}$	$y_{a<b}$
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

По таблице истинности одноразрядного компаратора (табл. 4.12) можно записать:

$$y_{a>b} = a\bar{b}, y_{a=b} = \overline{a\bar{b}} + \overline{ab} = \overline{a \oplus b}, y_{a<b} = \bar{a}b$$

и построить функциональную схему одноразрядного компаратора (рис. 4.54).

Многоразрядные числа сравнивают начиная со старших разрядов. На выход многоразрядного компаратора передают результат сравнения самых старших из несовпадающих разрядов.

Двоичный сумматор и АЛУ. Сумматором называется операционный узел ЦВМ, предназначенный для сложения двоичных чисел. При сложении двух одноразрядных чисел A и B возможны следующие комбинации:

$$0 + 0 = 0, 1 + 0 = 1, 0 + 1 = 1, 1 + 1 = 10,$$

где «+» — арифметическое сложение.

При $A = B = 1$ происходит перенос в старший разряд. Представляя числа A и B логическими переменными a_0 и b_0 , получим таблицу истинности (табл. 4.13), где c_1 — сигнал переноса, s_0 — сумма.

Таблица 4.13

a_0	b_0	s_0	c_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

По табл. 4.13 составим булевы функции:

$$c_1 = a_0 \wedge b_0, \quad s_0 = \bar{a}_0 \wedge b_0 \vee a_0 \wedge \bar{b}_0 = a_0 \oplus b_0.$$

Сумматор на 2 входа называется полусумматором (ПСМ). Функциональная схема ПСМ показана на рис. 4.55.

При сложении двух многоразрядных двоичных чисел полусумматор можно использовать только для младшего разряда. Во всех остальных разрядах складываются не 2, а 3 числа, поскольку может произойти перенос c_i со следующего за ним младшего разряда. Полный одноразрядный сумматор (ПОС) имеет 3 входа: a_i , b_i , c_i и два выхода s_i и c_{i+1} .

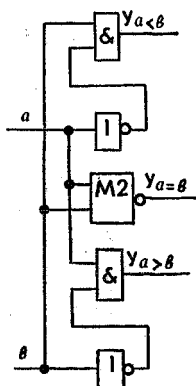


Рис. 4.54

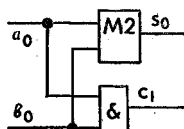


Рис. 4.55

ПОС можно, например, построить на двух ПСМ (рис. 4.56, а). УГО ПОС приведено на рис. 4.56, б. Закон функционирования ПОС показан в табл. 4.14.

Таблица 4.14

a_i	b_i	c_i	p_i	g_i	r_i	s_i	c_{i+1}
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	1	0	0	1	0
0	1	1	1	0	1	0	1
1	0	0	1	0	0	1	0
1	0	1	1	0	1	1	0
1	1	0	0	1	0	0	1
1	1	1	0	1	0	1	1

Из рис. 4.56, а следует $s_i = c_i \oplus p_i = c_i \oplus a_i \oplus b_i$. Если нанести на карту Карно — Вейча значения c_{i+1} из табл. 4.14, то после минимизации можно получить

$$c_{i+1} = a_i \wedge b_i \vee a_i \wedge c_i \vee b_i \wedge c_i.$$

Сумматор двух 4-разрядных двоичных чисел с последовательным переносом показан на рис. 4.57. Пере-

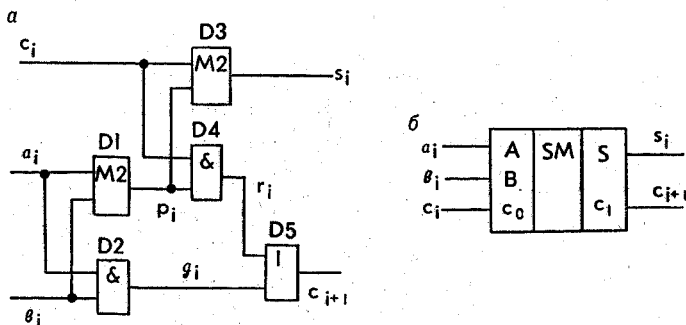


Рис. 4.56

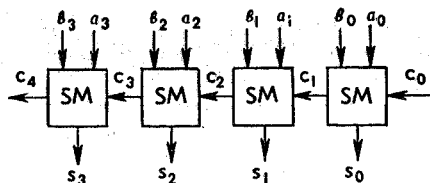


Рис. 4.57

носы c_4, c_3, c_2, c_1 вырабатываются во времени последовательно друг за другом, что приводит к увеличению времени суммирования многоразрядных чисел. Рассмотрим возможность реализации параллельного переноса.

Из рис. 4.56 можно записать

$$c_{i+1} = g_i \vee r_i = g_i \vee p_i \wedge c_i. \quad (4.8)$$

Сигнал g_i вырабатывается тогда, когда в данном разряде перенос образуется из-за комбинации входных переменных a_i и b_i . Поэтому g_i называют *функцией генерации* (или образования) *переноса*.

Сигнал p_i показывает, передается ли полученный в младшем разряде сигнал переноса c_i на выход c_{i+1} . Поэтому p_i называется *функцией распространения переноса*.

Пользуясь выражением (4.8), выведем формулы для сигналов переноса:

$$c_1 = g_0 \vee p_0 \wedge c_0; \quad (4.9)$$

$$c_2 = g_1 \vee p_1 \wedge c_1. \quad (4.10)$$

Подставив выражение (4.9) в (4.10), получим

$$c_2 = g_1 \vee p_1 \wedge g_0 \vee p_1 \wedge p_0 \wedge c_0. \quad (4.11)$$

Аналогично

$$c_3 = g_2 \vee p_2 \wedge c_2 = g_2 \vee p_2 \wedge g_1 \vee p_2 \wedge p_1 \wedge g_0 \vee \vee p_2 \wedge p_1 \wedge p_0 \wedge c_0; \quad (4.12)$$

$$c_4 = g_3 \vee p_3 \wedge c_3 = g_3 \vee p_3 \wedge g_2 \vee p_3 \wedge p_2 \wedge g_1 \vee p_3 \wedge p_2 \wedge \wedge p_1 \wedge g_0 \vee p_3 \wedge p_2 \wedge p_1 \wedge p_0 \wedge c_0 = G \vee P \wedge c_0,$$

где $G = g_3 \vee p_3 \wedge g_2 \vee p_3 \wedge p_2 \wedge g_1 \vee p_3 \wedge p_2 \wedge p_1 \wedge g_0$, (4.13)
 $P = p_3 \wedge p_2 \wedge p_1 \wedge p_0$.

Из выражений (4.9) — (4.13) следует, что если из схемы ПОС вывести на выход промежуточные величины

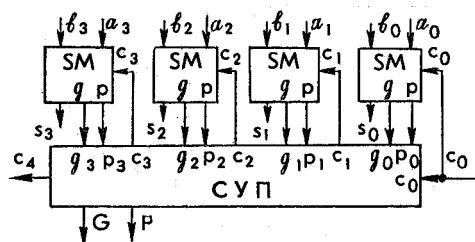


Рис. 4.58

g_i , p_i , т. е. применительно к рис. 4.56, а элементы $D4$ и $D5$ не ставить, а при тех же входах a_i , b_i , c_i выходами сделать s_i , p_i , g_i , то все переносы c_1 , c_2 , c_3 , c_4 будут получены одновременно (параллельно) через время задержки распространения сигнала на двух элементах И, ИЛИ. Четырехразрядный сумматор с параллельным переносом показан на рис. 4.58, где СУП — схема ускоренного переноса, реализующая булевы функции (4.9), (4.11), (4.12), (4.13).

Арифметико-логическое устройство (микросхема К155ИП3) содержит подобную схему ускоренного переноса. СУП выполняется также и в виде отдельной микросхемы К155ИП4, что позволяет, используя сигналы G и P , строить 16-разрядный сумматор из 4-разрядных секций, одна из которых приведена на рис. 4.58. Секции

(рис. 4.58) можно также включать последовательно, подключая c_4 к c_0 следующего более старшего сумматора.

На рис. 4.59 приведено УГО АЛУ К155ИП3. АЛУ выполняет 16 логических и 16 арифметических операций над двумя 4-разрядными словами $A = A_3A_2A_1A_0$ и $B = B_3B_2B_1B_0$.

АЛУ может работать в положительной и отрицательной логике. Микросхема К155ИП3 имеет следующие входы и выходы: c_0 — вход переноса; s_0, s_1, s_2, s_3 — управляющие или селектирующие входы выбора функции; M — вход «Режим работы»: при $M=1$ выполняются логические операции, при $M=0$ — арифметические; F_0-F_3 — четырехразрядный код результата арифметических или логических операций; c_4 — выход переноса; P — выход распространения переноса; G — выход образования (генерации) переноса; k — выход сравнения (выход компаратора): при $s_3=0, s_2=1, s_1=1, s_0=0$ и $A=B$ $k=1$.

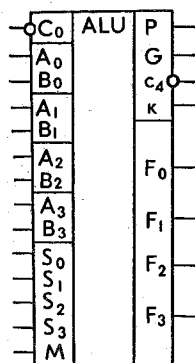


Рис. 4.59

В качестве примера в табл. 4.15 показано несколько арифметических и логических операций, выполняемых в положительной логике.

Таблица 4.15

Номер комбинации	Выбор функции $s_3s_2s_1s_0$	$M=1$. Логические функции	$M=0$. Арифметические функции	
			$\bar{c}_q = 1$	$\bar{c}_0 = 0$
0	0000	\bar{A}	A	$A+1$
1	0001	$A \vee B$	$A+B$	$(A+B)+1$
2	0010	$\bar{A} \wedge B$	$A+B$	$(A+B)+1$
и т. д.				

Регистр. Регистром называется операционный узел ЦВМ, представляющий упорядоченную последовательность запоминающих элементов со схемой управления. Регистр предназначен в основном для хранения и сдвига слова информации.

Рассмотрим несколько примеров схем регистров (для простоты трехразрядных).

На рис. 4.60, *а*, *б* соответственно показаны функциональная схема 3-разрядного регистра с *параллельным приемом и выдачей информации* и его УГО. Запоминающими элементами (рис. 4.60, *а*) служат синхронные D-триггеры, управляемые уровнем (см., например, рис. 4.36). При $C = 1$ регистр работает в режиме пере-

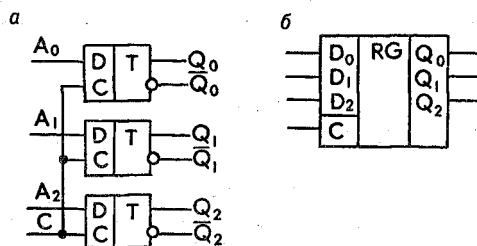


Рис. 4.60

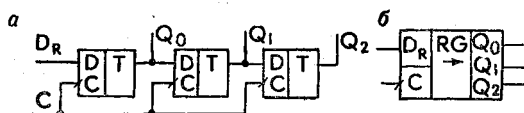


Рис. 4.61

дачи входной информации на выход. В этом случае под-держивается равенство $Q_i = A_i$. С появлением среза импульса по входу C , т. е. $C = 1 \rightarrow 0$, регистр переходит в режим хранения и выдачи записанной информации, т. е. $Q_i^{n+1} = A_i^n$. Изменения A_i при $C = 0$, согласно принципу работы синхронных триггеров, не приводят к изменению выходных сигналов Q_i . Обычно регистры имеют также вход установки в ноль R , по которому все триггеры могут быть переведены в нулевое состояние (на рис. 4.60 не показан).

На рис. 4.61, *а* приведен пример 3-разрядного *сдвигающего регистра*, построенного на D-триггерах с динамическим синхронизирующим входом (см., например, рис. 4.38), его УГО — на рис. 4.61, *б*. D_R — вход ввода информации при сдвиге вправо* (от Q_0 к Q_2), C — вход синхронизации. Работа регистра сдвига отражена в табл. 4.16, где № C — порядковый номер синхронимпульса, черточкой «—» показано произвольное состояние триггеров (0 или 1), в котором они находились до при-

* Индекс R от англ. right — правый.

хода синхроимпульсов, а стрелки указывают направления передачи информации.

В реверсивных регистрах информация может сдвигаться как вправо, так и влево. Последовательный и параллельный вводы информации можно реализовать и в одном регистре.

Таблица 4.16

№ С	D_R	Q_0	Q_1	Q_2
1	1	1	—	—
2	0	0	1	—
3	1	1	0	1

Схема 3-разрядного универсального сдвигающего регистра с параллельным вводом информации приведена на рис. 4.62. Регистр имеет дополнительный управляющий вход V . Когда $V=1$, регистр работает в режиме параллельного ввода информации, т. е. информация со входов A_0, A_1, A_2 с приходом $C=0 \rightarrow 1$ записывается одновременно (параллельно) во все разряды регистра, т. е. $Q_0=A_0, Q_1=A_1, Q_2=A_2$. При $V=0$ регистр работает в режиме сдвига вправо. С приходом $C=0 \rightarrow 1$ информация со входа D_R записывается в нулевой разряд регистра (т. е. $Q_0^{n+1}=D_R^n$), а ранее записанная в момент времени t_n информация сдвигается вправо ($Q_1^{n+1}=Q_0^n, Q_2^{n+1}=Q_1^n$). Если выполнить соединения, показанные на рис. 4.62 пунктиром, т. е. соединить Q_1 с A_0, Q_2 с A_1 , то при $V=1$ регистр будет работать в режиме сдвига информации влево (с приходом $C=0 \rightarrow 1$). При этом последовательный ввод информации осуществляется по входу A_2 , который служит входом D_L последовательного ввода информации при сдвиге влево*: $Q_2^{n+1}=A_2^n, Q_1^{n+1}=Q_2^n, Q_0^{n+1}=Q_1^n$. Проводя соединения, показанные на рис. 4.62 пунктиром, ликвидируем режим параллельного ввода информации, заменяя его на сдвиг влево. Можно построить схему регистра, который будет иметь все три режима: оба сдвига и параллельный ввод информации без необходимости проводить соединения типа Q_2-A_1, Q_1-A_0 .

Счетчики и пересчетные устройства. Счетчик — операционный узел ЦВМ, обеспечивающий хранение слова информации и выполнение над ним микрооперации

* Индекс L от англ. left — левый.

счета. Микрооперация счета состоит в изменении значения слова (состояния счетчика) на 1. Счетчик, на котором реализуется микрооперация вида $C := C + 1$, называется *суммирующим*. Если на счетчике реализуется микрооперация $C := C - 1$, то это *вычитающий* счетчик. Счетчик, на котором реализуются обе указанные микрооперации, является *реверсивным*.

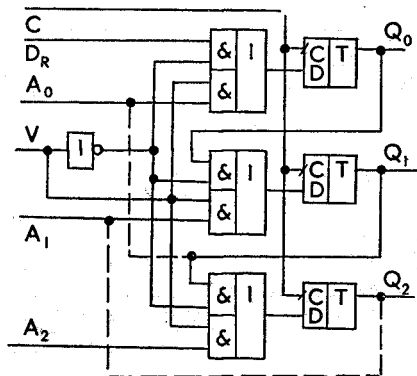


Рис. 4.62

Пересчетным устройством называется операционный узел ЦВМ, обеспечивающий хранение слова информации и выполнение над ним микрооперации произвольной смены состояния. Вследствие этого определения «суммирующий», «вычитающий», «реверсивный» неприменимы к пересчетным устройствам.

Счетчики и пересчетные устройства применяются для образования адресов слов в запоминающих устройствах (ЗУ) и адресов команд в устройствах управления (УУ), а также выработки специальных последовательностей управляющих сигналов, подсчета числа циклов сложения и вычитания, построения распределителей импульсов и т. п. В процессе работы счетчик последовательно изменяет свои состояния. Длину списка разрешенных состояний называют *модулем счета*, либо *основанием счета*, или *емкостью счетчика k*. Одно из возможных состояний счетчика принимается за начальное. Если счетчик начал работать от начального состояния, то каждый входной сигнал, кратный k , снова устанавливает счетчик в начальное состояние, а на выходе счетчика

появляется сигнал k -ичного переноса p (в суммирующем счетчике) или заема (в вычитающем счетчике).

Двоичный счетчик имеет модуль счета 2^N , где N — число двоичных разрядов; десятичный счетчик имеет модуль 10^m , где m — число десятичных разрядов.

Соответствие между числом входных импульсов Q^\pm трехразрядного двоичного счетчика и значениями выходных двоичных переменных $Q_2Q_1Q_0$ с весами соответственно $2^2, 2^1, 2^0$ показано в табл. 4.17.

Таблица 4.17

Q^+	Q_2	Q_1	Q_0	Q^-
	2^2	2^1	2^0	
1	0	0	0	8
2	0	0	1	7
3	0	1	0	6
4	0	1	1	5
5	1	0	0	4
6	1	0	1	3
7	1	1	0	2
8	0	1	1	1
0	0	0	0	0

Просматривая таблицу сверху вниз, можно отметить две закономерности для суммирующего счетчика (Q^+): 1) значение переменной Q_i изменяется тогда, когда переменная в соседнем младшем разряде переходит из состояния 1 в состояние 0; 2) значение Q_i изменяется при поступлении очередного импульса в том случае, если переменные во всех младших разрядах Q_{i-1}, \dots, Q_0 находятся в состоянии 1. Например, в состояниях счетчика $Q_2Q_1Q_0$ 011 и 111 $Q_1 = Q_0 = 1$, с приходом очередного импульса (4-го или 8-го) Q_1 изменяется с 1 на 0, что приводит к изменению Q_2 на Q_2 , т. е. после 011 будет 100, а после 111 — 000.

Первая закономерность позволяет реализовать счетчик асинхронного типа (с последовательным переносом), когда переключение последующего триггера осуществляется только с выхода предыдущего, а входные счетные импульсы поступают только на вход младшего (нулевого) триггера. В этом случае из состояния 111 в 000 с приходом 8-го импульса триггеры переключаются после-

довательно ($111 \rightarrow 110 \rightarrow 100 \rightarrow 000$) за 3τ , где τ — время переключения одного триггера.

Вторая закономерность позволяет строить *синхронный счетчик*, когда входные счетные импульсы поступают на все триггеры и их переключение осуществляется одновременно за время τ в зависимости от состояния предшествующих младших разрядов.

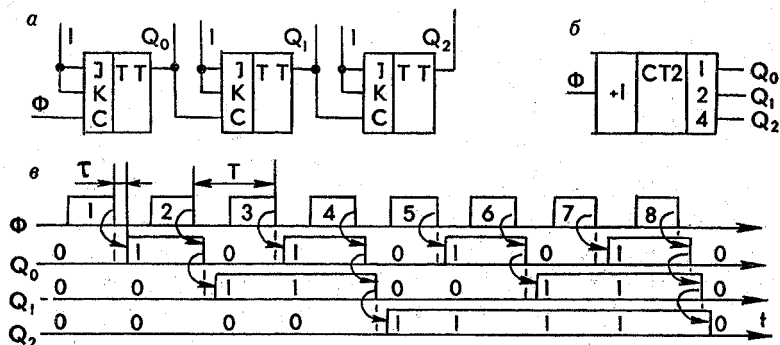


Рис. 4.63

Просматривая табл. 4.17 снизу вверх, можно также заметить две закономерности работы вычитающего счетчика (число входных импульсов Q^-): 1) значение выходной переменной Q_i изменяется тогда, когда Q_{i-1} переходит из состояния 0 в состояние 1; 2) значение переменной Q_i изменяется при поступлении очередного импульса счета в том случае, если во всех младших разрядах Q_{i-1}, \dots, Q_0 переменные находятся в состоянии 0, т. е. возможно построение как асинхронного, так и синхронного счетчика.

Рассмотрим несколько примеров построения счетчика.

На рис. 4.63, *а* приведена функциональная схема асинхронного двоичного трехразрядного суммирующего счетчика на двухступенчатых JK-триггерах (например, на микросхемах K155ТВ1, рис. 4.41), а на рис. 4.63, *б* — его УГО. Этот счетчик можно также назвать счетчиком с *последовательным переносом*. Временная диаграмма работы данного счетчика, на вход которого поступают счетные импульсы, приведена на рис. 4.63, *в*. Стрелками показаны перепады напряжения, приводящие к изменению сигналов Q_i . На последующих рисунках задержки на срабатывание триггеров показывать не будем.

Каждый разряд счетчика (рис. 4.63, *a*) представляет собой асинхронный Т-триггер (счетный триггер, см. рис. 4.44). Кроме основной функции — подсчета импульсов, счетчик обеспечивает деление частоты следования импульсов. Если частоту входных импульсов обозначить через $f = 1/T$, где T — период следования импульсов, то на выходе Q_0 сигнал изменяется с частотой $f/2$, на

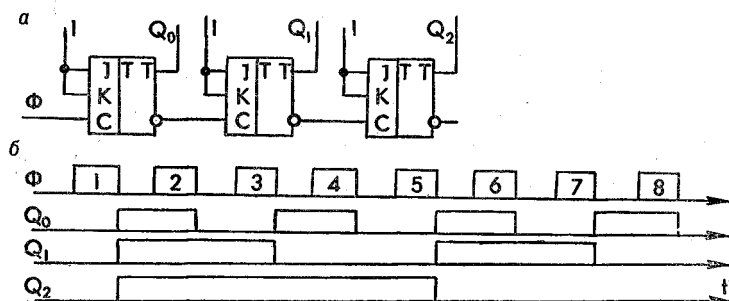


Рис. 4.64

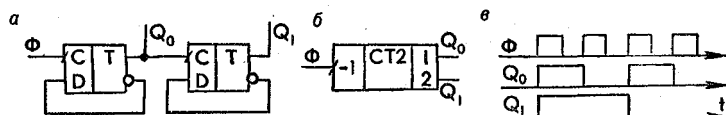


Рис. 4.65

выходе Q_1 — с частотой $f/4$, на выходе Q_2 — с частотой $f/8$. В то время, когда на выходе Q_{i-1} сигнал изменяется с уровня логической 1 в уровень логического 0, на выходе \bar{Q}_{i-1} происходит изменение сигнала из 0 в 1. Поэтому если соединить входы C_i с выходами \bar{Q}_{i-1} , то получим вычитающий асинхронный двоичный счетчик (рис. 4.64, *a*), временная диаграмма работы которого отражена на рис. 4.64, *б*.

На рис. 4.65, *a* показан пример функциональной схемы двухразрядного вычитающего двоичного счетчика на D-триггерах, управляемых фронтом, на рис. 4.65, *б* — его УГО, а на рис. 4.65, *в* — временная диаграмма работы. Так как Т-триггер на D-триггерах переключается по перепаду входного сигнала $0 \rightarrow 1$, то соединением C_i с Q_{i-1} получается вычитающий счетчик, а соединением C_i с \bar{Q}_{i-1} — суммирующий.

На рис. 4.66 дана схема 4-разрядного двоичного синхронного суммирующего счетчика с *параллельным (одновременным) переносом*. Это самый быстрый вид переноса, так как сигналы переноса на входах T всех триггеров Q_1, Q_2, Q_3 формируются одновременно и переключение всех триггеров $Q_0 — Q_3$ осуществляется одновременно за время t .

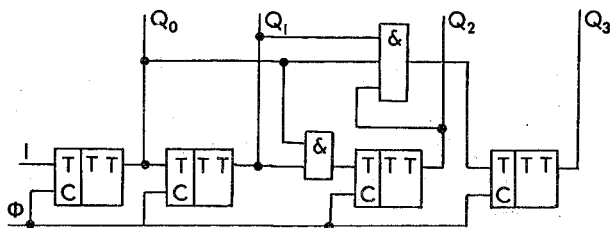


Рис. 4.66

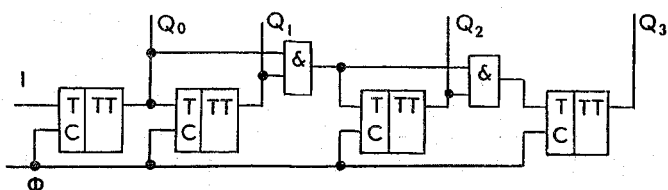


Рис. 4.67

Как видно из рис. 4.66, первая схема И имеет 2 входа, вторая — 3 входа. При увеличении числа разрядов следующая схема И будет иметь 4 входа и т. д. Построение многоразрядного счетчика с параллельным переносом идет до тех пор, пока хватает входов логических элементов И, а затем переходят к схеме со сквозным или последовательным переносом.

На рис. 4.67 показана схема 4-разрядного синхронного двоичного суммирующего счетчика со *сквозным переносом*. Этот вид переноса более быстродействующий, чем последовательный, так как перенос от разряда к разряду идет одновременно с процессом переключения триггеров и опережает во времени процесс переключения триггеров. Однако сквозной перенос по быстродействию уступает параллельному, так как он распространяется через последовательную цепочку элементов И, задержки на которых суммируются.

На рис. 4.68, а, показана функциональная схема 2-разрядного реверсивного двоичного счетчика, где p_1 — выход переноса, p_2 — выход заема. УГО этого счетчика и временная диаграмма работы соответственно приведены на рис. 4.68, б, в.

При входном сигнале $\Phi_2 = 0$ со входом C триггера F_1 соединен выход Q_0 триггера F_0 , что соответствует

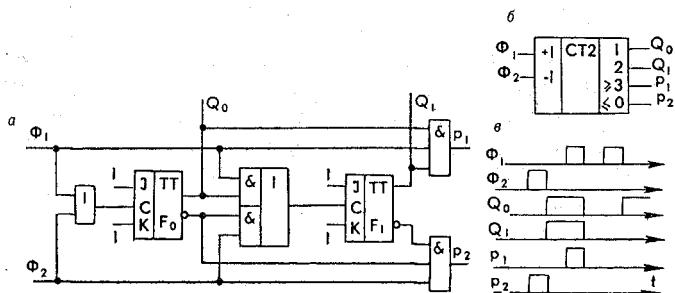


Рис. 4.68

работе суммирующего счетчика по входу суммирования (см. рис. 4.63), а при входном сигнале $\Phi_1 = 0$ с этим же входом C соединен выход \bar{Q}_0 триггера F_0 , что соответствует работе вычитающего счетчика по входу вычитания (см. рис. 4.64). Импульсы Φ_1 и Φ_2 должны быть сдвинуты относительно друг друга на время, достаточное для срабатывания триггера, что обеспечит правильную работу счетчика.

Прежде проводился анализ работы готовых схем, а теперь рассмотрим пример синтеза пересчетного устройства (ПСУ) 1—3—2 на D-триггерах с динамическим синхронизирующим входом (микросхема К155ТМ2). Заданы три состояния 1, 3, 2, т. е. $k=3$. Один триггер дает $2^1=2$ состояния, что меньше необходимых трех, 2 триггера имеют $2^2=4$ состояния, что больше трех. Выбираем 2 триггера.

Составим таблицу функционирования пересчетного устройства (табл. 4.18). Нам даны состояния в десятичной системе счисления $Q_{10} = 1, 3, 2$, которые в двоичном коде Q_2Q_1 будут эквивалентны 01, 11, 10. Необходимо определить, какие наборы подавать на входы D_2D_1 . Для того чтобы ПСУ перешло из состояния 01 в 11, необходимо в состоянии 01 на входах D_2D_1 подготовить код 11 (согласно логическому уравнению D-триггера $Q_{n+1} =$

Таблица 4.18

Q	Q_2	Q_1	D_2	D_1
1	0	1	1	1
3	1	1	0	0
2	1	0	0	1
0	0	0	0	1

$= D_n$). Аналогично заполняем остальные строки в соответствии со стрелками табл. 4.18. При включении питания схема может оказаться в состоянии 00. Если в это время на входах D_2D_1 также будет 00, то, несмотря на входные импульсы, схема «застрянет» в состоянии 00. Поэтому при $Q_2 = Q_1 = 0$ предусмотрим переход в одно из разрешенных состояний, например 01 (см. табл. 4.18). Затем нанесем единичные значения D_2 и D_1 на карты Карно — Вейча (рис. 4.69, а, б), проведем контуры и запишем булевы функции:

$$D_2 = Q_1, D_1 = \bar{Q}_2 + \bar{Q}_1 = \overline{\overline{\bar{Q}_2 + \bar{Q}_1}} = \overline{Q_2 Q_1}.$$

По этим функциям составим функциональную схему ПСУ (рис. 4.69, в). На рис. 4.70 показан граф ПСУ.

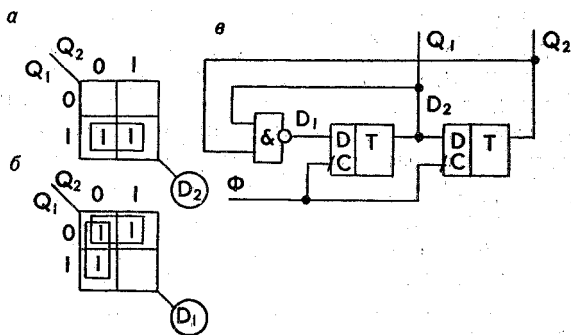


Рис. 4.69

Рассмотрим пример синтеза этого же ПСУ 1—3—2 на JK-триггерах (микросхема К155ТВ1). Количество триггеров то же. Составим таблицу функционирования ПСУ (табл. 4.19). Для заполнения значений J и K используем граф JK-триггера (рис. 4.71). На дугах показаны входные сигналы J и K , по которым осуществляется переход,

Таблица 4.19

Q_2	Q_1	J_2	K_2	J_1	K_1
0	1	1	*	*	0
1	1	*	0	*	1
1	0	*	1	1	*
0	0	*	*	*	*

звездочкой «*» отмечено безразличное значение (0 или 1).

Например из 0 в 1 JK-триггер перейдет при входных сигналах $J = 1$, $K = 0$ или $J = 1$, $K = 1$ (см. табл. 4.3 и 4.7), т. е. обязательно $J = 1$, а K может быть 0 или 1.

На примере Q_2 в табл. 4.19 стрелками показаны переходы второго триггера. Для перехода из 0 в 1 необходимо в первой строке записать $J_2 = 1$, $K_2 = *$. Для перехода из 1 в 1 необходимо во второй строке записать $J_2 = *$, $K_2 = 0$, по такой же методике заполняется третья строка. Аналогично заполняются строки для первого триггера.

Для четвертой строки возможны 2 варианта заполнения. Если предположить, что при включении питания будет выполнена предварительная установка по входам \bar{R} , \bar{S} в одно из разрешенных состояний, то в последней строке для J и K можно поставить *. Если же не будет

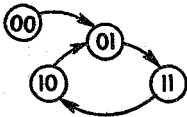


Рис. 4.70

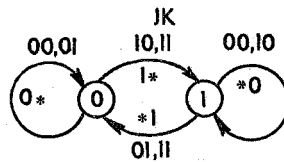


Рис. 4.71

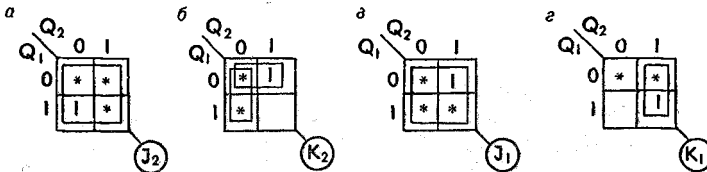


Рис. 4.72

предварительной установки, то, как и в примере для ПСУ на D-триггерах, необходимо предусмотреть переход из 00 в одно из разрешенных состояний.

Если нанести J_2K_2 , J_1K_1 на карты Карно — Вейча (рис. 4.72), то можно записать булевы функции для этих входов, а по ним составить функциональную схему ПСУ (рис. 4.73).

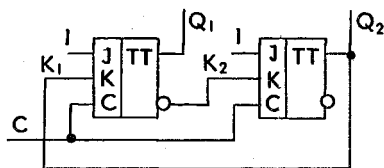


Рис. 4.73

4.4. Полупроводниковые БИС запоминающих устройств

Общие сведения. Классификация наиболее распространенных БИС ЗУ приведена на рис. 4.74. Интегральные схемы полупроводниковых ЗУ позволяют создавать устройства памяти с большой информационной емкостью. По режиму занесения информации полупроводниковые ЗУ делятся на *оперативные запоминающие устройства* (ОЗУ), которые по ГОСТ 2.743—82 обозначаются RAM*, и *постоянные запоминающие устройства* (ПЗУ).

По технологии изготовления ЗУ подразделяют на *биполярные* и *униполярные***. Обычно биполярные ЗУ имеют значительно большее быстродействие, но меньшую плотность упаковки элементов по сравнению с униполярными.

По уровням входных и выходных сигналов ЗУ любого типа изготавливаются совместимыми с логическими элементами одной из трех стандартных систем: ЭСЛ, ТТЛ, КМОП. ЗУ на основе И²Л рассчитаны на работу с микросхемами ТТЛ или реже ЭСЛ, а ЗУ на основе *p*-МОП и *n*-МОП совместимы с микросхемами ТТЛ. ЗУ на основе КМОП совместимы по входу с логическими элементами КМОП, а по выходу — с ТТЛ элементами. При использовании в устройствах ТТЛ уровней на входе КМОП ЗУ обычно требуются схемы преобразователей уровней ТТЛ — КМОП.

* От англ. random-access memory — запоминающее устройство с произвольной выборкой (ЗУПВ), оперативное запоминающее устройство.

** Сокращение на рис. 4.74 *n*-ЛИЗМОП — лавинно-инжекционный МОП-транзистор *n*-типа с изолированным затвором.

На рис. 4.74 буквами РУ, РЕ, РТ, РР, РФ отмечены условные обозначения вида изделия по функциональному назначению. Пример обозначения ИС: КР565РУ1А — микросхема общетехнического применения, в пластмассовом корпусе, полупроводниковая, серия 565, оперативное ЗУ, разработка 1, типонаименование А.

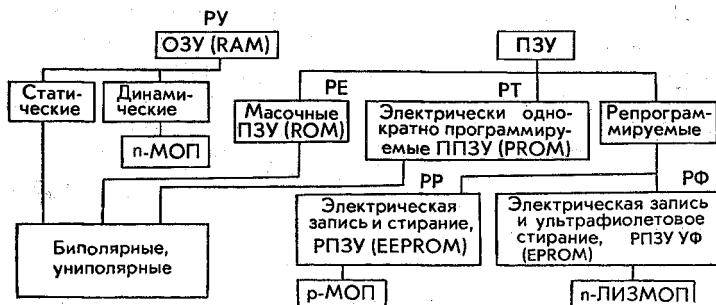


Рис. 4.74

Оперативные запоминающие устройства. ОЗУ используются для введения в процессор текущих результатов или данных, полученных в процессе работы. По способу хранения информации ОЗУ подразделяют на статические и динамические.

Элементы памяти статических ЗУ представляют собой бистабильные элементы. Они управляются потенциальными сигналами. Считывание информации осуществляется без ее разрушения.

В динамических ОЗУ в качестве запоминающего элемента (ЗЭ) используется конденсатор, в котором информация хранится в виде заряда. Заряд на запоминающем конденсаторе с течением времени уменьшается за счет токов утечки. Для восстановления заряда требуется периодическая подзарядка (регенерация) накопительного конденсатора, т. е. для управления требуются импульсно-потенциальные сигналы. При считывании информация, как правило, разрушается, но за счет регенерации снова восстанавливается. Динамические ОЗУ выполняются на n-МОП-транзисторах, имеют высокую степень интеграции, эффективны для построения ОЗУ большой емкости.

Постоянные запоминающие устройства. ПЗУ — устройства, из которых можно считывать только зара-

нее записанную информацию. Они служат для генерации кода какой-либо программы или данных, которые будут часто использоваться, что избавляет от необходимости загружать программу каждый раз заново. Информация в ПЗУ, в отличие от ОЗУ, записывается на кристалле с изменением его физических свойств, поэтому при отключении питания она сохраняется.

В наиболее простых масочных ПЗУ информация записывается при их изготовлении на заводе заменой одного из фотошаблонов слоя коммутации.

Более удобны для пользователя электрически программируемые ПЗУ (ППЗУ), однако они позволяют только однократно записывать информацию. Запись производится у потребителя на специальном стендовом оборудовании путем разрушения элементов структуры ППЗУ под действием приложенного электрического напряжения или тока. Разрушаемыми элементами структуры могут быть проводящие плавкие перемычки, а также тонкий слой диэлектрика или $p-n$ -переходы. К недостаткам ППЗУ относятся самовосстановление перемычек и достаточно большое время на процесс программирования (несколько сотен миллисекунд на каждую перемычку или диод).

Репрограммируемые ПЗУ (РПЗУ) позволяют многократно записывать и стирать информацию. РПЗУ на МНОП (металл-нитрид-оксид-полупроводник) и МАОП (металл-алунд-оксид-полупроводник) позволяют осуществлять электрическую запись и стирание на специальном стендовом оборудовании. В РПЗУ УФ стирание информации осуществляется ультрафиолетовым облучением через входное окно корпуса ИС, а запись — электрическими сигналами.

РПЗУ позволяют сохранять информацию при отключении питания, например РПЗУ на МНОП-транзисторах К1601РР1 в течение 5000 ч, РПЗУ УФ К573РФ1 — 15 000 ч.

Разработчики микроэлектронной аппаратуры применяют БИС РПЗУ, РПЗУ УФ для отладки и проверки правильности используемых программ, а затем, убедившись в их правильности, заменяют РПЗУ, РПЗУ УФ дешевыми и быстродействующими БИС ПЗУ или БИС ППЗУ. Недостатком РПЗУ является их низкое быстродействие.

Основные параметры ЗУ. Информационная емкость N — это максимальный объем хранимой

информации (бит, К бит*), определяемый числом ЗЭ в накопителе. При одинаковой емкости БИС ЗУ могут иметь различную организацию выборки. Например, для $N = 4096 - 4096 \times 1$ (4096 одноразрядных слов), 1024×4 (1024 четырехразрядных слов), 512×8 , 256×16 .

Быстродействие БИС ЗУ характеризуется двумя основными параметрами: *временем выборки адреса* $t_{в.а}$ и *временем цикла записи* $t_{ц.зп}$ (считывания $t_{ц.сч}$).

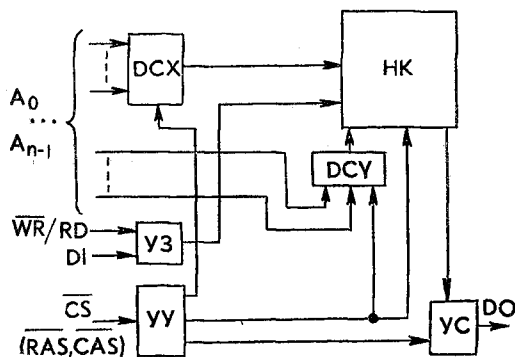


Рис. 4.75

Величина $t_{в.а}$ — интервал времени между моментом подачи сигнала выборки и появлением информации на выходе БИС; время цикла $t_{ц}$ — интервал времени между началами (окончаниями) сигналов на одном из управляющих входов, в пределах которого БИС выполняет одну из функций (запись $t_{ц.зп}$, считывание $t_{ц.сч}$).

Кроме того, БИС ЗУ характеризуются удельной потребляемой мощностью P_0 , т. е. мощностью потребляемой БИС, отнесенной к ее информационной емкости, или током $I_{пот}$, потребляемым от источника питания.

Схемотехника ЗУ. На рис. 4.75 приведена типовая структурная схема полупроводниковых БИС ЗУ (ОЗУ, ППЗУ, РПЗУ). Она включает накопитель НК (матрица ЗЭ), дешифратор строк DCX, дешифратор столбцов DCY, устройство записи Y3, устройство считывания YC, устройство управления YU. Входные информационные сигналы DI поступают в устройство записи Y3, которое служит для записи информации в ЗЭ накопителя. Выход-

* 1 К бит = 2^{10} бит = 1024 бит.

ные информационные сигналы DO считываются из БИС ЗУ через устройство считывания УС. Управляющие сигналы \overline{CS} , RAS, \overline{CAS} , $\overline{WR/RD}$ поступают в УУ и УЗ и определяют режим работы БИС ЗУ (запись, хранение, считывание информации).

Входы и выходы ЗУ имеют следующие обозначения: DO (от англ. data output) — данные выходные; DI (data

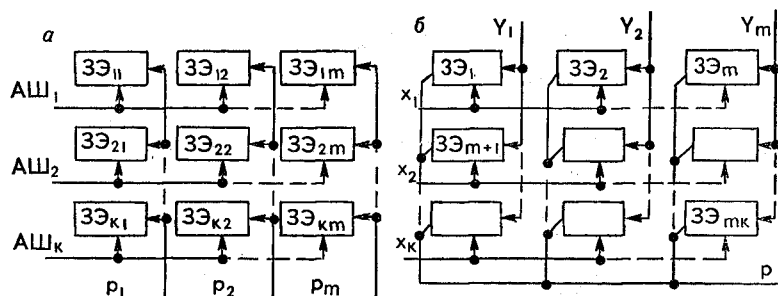


Рис. 4.76

input) — данные входные; \overline{CS} (chip select) — выбор микросхемы; \overline{WR} (write) — писать; \overline{RD} (read) — читать; RAS (row address select) — выбор адреса строки (строб адреса строки); \overline{CAS} (column address select) — выбор адреса столбца (строб адреса столбца). Входной сигнал $\overline{WR/RD}$ (запись-считывание) на входе микросхемы означает, что нулем (\overline{WR}) осуществляется режим записи, а единицей по этому же входу — режим чтения (\overline{RD}).

Адресные сигналы A_0, \dots, A_{n-1} поступают на схемы дешифрации DCX, DCY, которые определяют, к какому элементу памяти накопителя производится обращение в соответствии с заданным кодом адреса. Выходные схемы связаны с устройством считывания УС, которое служит для усиления считанной из накопителя информации и передачи ее на выход DO. Во многих случаях выходные схемы имеют возможность передачи трех логических состояний: 1, 0 и состояния высокого сопротивления на выходе R_{off} , что облегчает объединение по выходу БИС ЗУ в системах с шинной организацией передачи данных. Типовая структурная схема ПЗУ отличается лишь отсутствием устройства записи УЗ.

Рассмотрим две упрощенные структурные схемы накопителя (матрицы 3Э) статического ЗУ на биполярных транзисторах (рис. 4.76).

По принципу построения накопителя информации БИС ЗУ бывают с *однокоординатной выборкой* (со словарной организацией, рис. 4.76, а) и *двухкоординатной выборкой* (с матричной организацией, рис. 4.76, б). При словарной организации используется одна ступень дешифрации (только DCX). Выходами DCX служат адресные шины $АШ_1, \dots, АШ_k$ (строки). Одна строка образует слово из

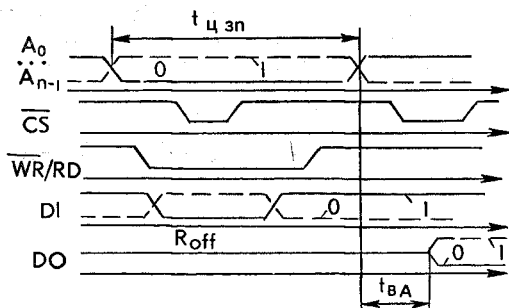


Рис. 4.77

m разрядов. При наличии на одной из шин, например первой, единичного сигнала ($АШ_1 = 1$) состояние каждого ЗЭ первого слова ($ЗЭ_{11}, ЗЭ_{12}, \dots, ЗЭ_{1m}$) считывается в режиме считывания по разрядным шинам P_1, \dots, P_m . Если необходимо записать информацию в первую строку (первое слово), то в режиме записи на разрядные шины p_1, \dots, p_m подаются сигналы 1 или 0, которые при $АШ_1 = 1$ попадут на каждый из ЗЭ первой строки. Организация такого запоминающего устройства $n \cdot m$ -разрядных слов, емкость $N = n \cdot m$.

При двухкоординатной выборке применяют две ступени дешифрации (DCX и DCY). Обычно используется квадратная матрица $k = m$, где $x_1, \dots, x_k, y_1, \dots, y_m$ — соответственно выходы DCX и DCY. При наличии только $x_1 = y_1 = 1$ в режиме считывания будет выбран только ЗЭ₁. Его состояние (0 или 1) считывается по общей разрядной шине p . Чтобы записать 1 в этот же элемент ЗЭ₁, необходимо при $x_1 = y_1 = 1$ в режиме записи на разрядную шину p подать 1. Организация такого ЗУ $m \cdot k$ одно-разрядных слов, емкость $N = m \cdot k$.

На рис. 4.77 показана типовая временная диаграмма работы полупроводниковых БИС ОЗУ, ППЗУ, РПЗУ. Двойными линиями (сплошной и пунктирной) показаны

возможные формы сигналов на соответствующих выводах микросхемы. На выходах DO возможны 3 состояния: 0, 1 и высокое выходное сопротивление (R_{off}).

В качестве примера рассмотрим структурную схему программируемого запоминающего устройства КР556Т4А емкостью 1024 бит с организацией 256×4 (рис. 4.78, а). Его УГО изображено на рис. 4.78, б.

Это — БИС ТТЛ с диодами Шотки. Запись информации (программирование) производится потребителем путем пережигания нихромовых перемычек импульсом тока один раз во время эксплуатации схемы. Считывание производится при подаче логического 0 на входы \overline{CS}_1 и \overline{CS}_2 . Таблица истинности микросхемы К556РТ4А представлена в табл. 4.20, где M — любая комбинация сигналов \overline{CS}_1 , \overline{CS}_2 кроме 00.

Т а б л и ц а 4.20

\overline{CS}_1	\overline{CS}_2	$A_0...A_7$	$DO_0...DO_3$	Режим работы
M	M	\times	1	Хранение (не выбор)
0	0	А	Данные в прямом коде	Считывание

При считывании на входы A_0, \dots, A_7 подается код адреса A считываемого слова. Входной дешифратор DC выбирает одну из 32 строк запоминающей матрицы накопителя, содержащей 32 бита (восемь 4-разрядных слов). Для считывания одного из восьми слов, выбранных входным дешифратором DC , служит мультиплексор-селектор MS , управляемый тремя адресными входами A_0, \dots, A_2 . Код считанного слова через считывающие формираторы CF подается на внешнюю нагрузку. Выходы CF выполнены по схеме с открытым коллектором. Время выборки адреса $t_{в.а} \leq 70$ нс.

Структурную схему ПЗУ часто изображают более подробно. Так, входной дешифратор DC (рис. 4.78, в) показывают состоящим из адресных инверторов AI и дешифратора строк DC_x (см., например, рис. 4.46, а), мультиплексор-селектор MS (рис. 4.78, г) показывают либо как 4 мультиплексора 8—1 с общими адресными входами A_0, \dots, A_2 , либо состоящим из AI (рис. 4.78, д), выходного дешифратора DC_y (дешифратора столбцов) и селектора (см., например, рис. 4.50, а). В данном случае селек-

тор — это устройство, осуществляющее выборку 4 разрядов из 32 и являющееся одновременно разрядным формирователем. Считывающие формирователи иногда изображают так, как показано на рис. 4.78, е.

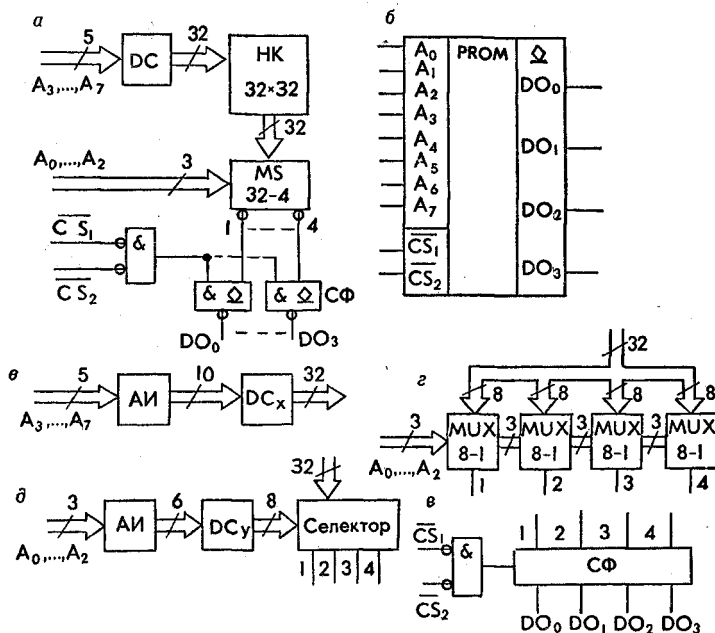


Рис. 4.78

На ПЗУ можно реализовать различные комбинационные схемы, работающие в соответствии с заданной таблицей истинности. Для этого адресные сигналы A_0, \dots, A_n рассматривают как логические переменные.

Недостатками ПЗУ являются необходимость занесения всей таблицы истинности, а также применение нескольких микросхем при большом числе входных переменных.

Часто только небольшое число строк в таблице истинности содержит единицы, а в остальных стоят нули (или наоборот) или в таблице обнаруживается какая-либо закономерность. В этом случае можно реализовать данную таблицу с помощью соответствующих булевых функций y_1, \dots, y_m . Исходя из дизъюнктивной нормальной формы (ДНФ) булевы функции можно записать в следующем виде (для наглядности взят простой пример):

$$y_1 = \overline{x_1}x_2 + x_1\overline{x_2}x_4 = X_1 + X_2;$$

$$y_2 = \overline{x_1}x_2 + x_1x_3x_4 = X_1 + X_3;$$

$$y_3 = x_1\overline{x_2}x_4 + x_1x_3x_4 + x_1x_2x_3 = X_2 + X_3 + X_4$$

и реализовать с помощью двух матриц: матрицы И (M_1), выходами которой будут X_i , и матрицы ИЛИ (M_2), выходами которой будут y_j (рис. 4.79).

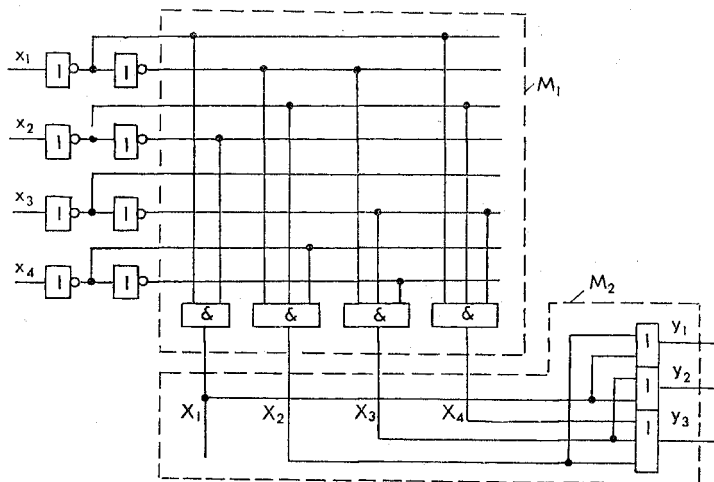


Рис. 4.79

Необходимые связи в матрицах M_1 и M_2 осуществляются с помощью простых соединений пересекающихся проводов. Подобные структуры (рис. 4.79) называются *программируемыми логическими матрицами* (ПЛМ) и имеют условное обозначение вида изделия по функциональному назначению РТ.

На рис. 4.80 показана упрощенная (без цепей программирования) функциональная схема ПЛМ 556РТ1. Она состоит из адресных формирователей (адресных инверторов) АФ, матрицы И на диодно-резисторных схемах с диодами Шотки, матрицы ИЛИ на эмиттерных повторителях (объединение эмиттеров дает функцию ИЛИ) и усилителя считывания с открытым коллектором. Если на входе разрешения выборки единица ($\overline{PB} = 1$), то все выходные транзисторы, подключенные к выходам F_0, \dots, F_7 , закрыты. При $\overline{PB} = 0$ на выходе F_0, \dots, F_7 нули, так как все перемычки до начала программирования

целые. Действительно, например, для выхода F_7 можно записать $F_7 = \overline{y_7}PB$, $y_7 = z_1z_2 + z_1\overline{z_2}$.

Так как $z_2 = 0$ (перемычка по входу z_2 целая, имеет небольшое сопротивление), то

$$y_7 = z_1 \cdot \overline{0} + z_1 \cdot 0 = \overline{z_1}.$$

Поскольку $z_1 = 0$ (низкий уровень со схем ИЛИ), то $y_7 = \overline{z_1} = 1$, а $F_7 = \overline{y_7} \cdot \overline{0} = \overline{y_7} = z_1 = 0$.

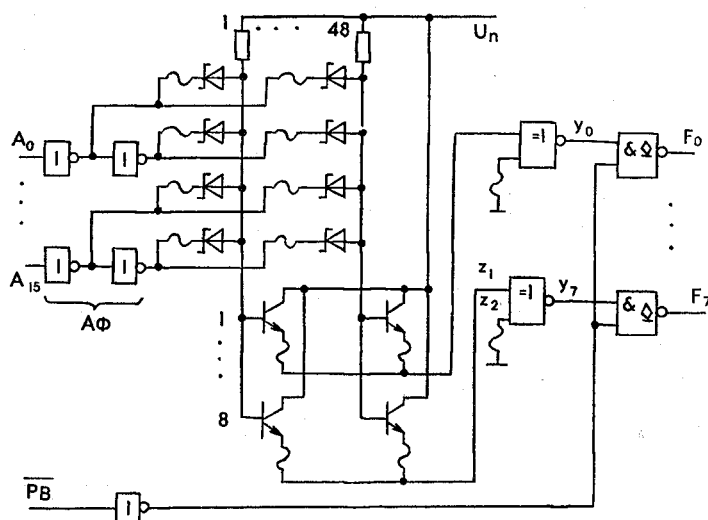


Рис. 4.80

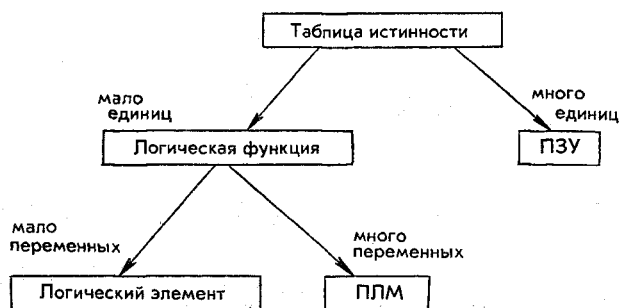


Рис. 4.81

Если переключить переключатель по входу z_2 , то z_2 будет равен 1 и

$$y_7 = z_1 \cdot \overline{1} + \overline{z_1} \cdot 1 = z_1,$$

т. е. возможно инвертирование выходного сигнала с помощью переключателя по входу z_2 .

Микросхема ПЛМ 556РТ1 имеет 16 адресных входов, 48 промежуточных шин и 8 выходов. Переключатели переключаются последовательно по одной в режиме программирования импульсами тока 40 мА. Различные способы реализации комбинационных схем приведены на рис. 4.81.

Контрольные вопросы и задания

1. Изобразите диодно-резисторные логические схемы И (ИЛИ) для положительных и отрицательных сигналов. От чего зависят и как определяются выходные уровни ДРЛ?

2. Перечислите достоинства и недостатки НСТЛ, РСТЛ, РКТЛ.

3. Для чего применяют схемы с открытым коллектором?

4. Нарисуйте схему И — НЕ ДТЛ. Как определяется помехоустойчивость схемы?

5. Нарисуйте схему И — НЕ ТТЛ. Объясните назначение всех компонентов схемы. Как рассчитать помехоустойчивость схемы?

6. Можно ли схемы ТТЛ объединять по выходу? Для чего нужны схемы с тремя состояниями выхода? Нарисуйте схему И — ИЛИ — НЕ ТТЛ и объясните, как она работает.

7. Чем объясняется высокое быстродействие ЭСЛ? Нарисуйте базовый элемент серии 500 и объясните его работу. Как рассчитать выходные уровни ЭСЛ? Какие логические функции получаются при объединении выходов двух элементов ЭСЛ между собой?

8. Объясните принцип работы схем И²Л, укажите их основное достоинство.

9. Сравните между собой различные схемы на полевых транзисторах. Приведите схемы И — НЕ, ИЛИ — НЕ на КМОП-транзисторах. Чему равны выходные уровни напряжений?

10. Приведите классификацию триггерных схем. Как определяется рабочая частота RS-триггера? Что такое неопределенное состояние?

11. Приведите функциональные схемы рассмотренных триггеров, их таблицы состояний.

12. Приведите схемы операционных узлов ЦВМ и поясните принципы их функционирования.

13. Приведите классификацию полупроводниковых БИС ЗУ. Расскажите об их назначении и структурной организации.

Упражнения

4.1. В схеме на рис. 4.6 при $R_n = 2R$, $E = 6$ В, $U_{до} = 0,7$ В определите выходное напряжение U_y для следующих значений входных напряжений: а) $U_{x1} = 0,2$ В, $U_{x2} = 3$ В; б) $U_{x1} = 3$ В, $U_{x2} = 5$ В; в) $U_{x1} = 3,6$ В, $U_{x2} = 5$ В; г) $U_{x1} = U_{x2} = 7$ В; д) $U_{x1} = -2$ В, $U_{x2} = 3$ В; е) $U_{x1} = U_{x2} = 0,4$ В. Какие диоды открыты, какие закрыты?

4.2. В схеме на рис. 4.9, а при $R_n = R$, $E = -5$ В, $U_{до} = 0,7$ В определите U_y при следующих значениях входных напряжений:

а) $U_{x1} = -0,2$ В, $U_{x2} = -3,6$ В; б) $U_{x1} = U_{x2} = -0,2$ В; в) $U_{x1} = -5$ В, $U_{x2} = -2,2$ В; г) $U_{x1} = 2,4$ В, $U_{x2} = -1$ В; д) $U_{x1} = U_{x2} = -6$ В.

4.3. Для схемы на рис. 4.10, а определите $U_{пор}^0$ при $R1 = R2$.

4.4. Для схемы на рис. 4.10, б при изменении входного напряжения в соответствии с рис. 4.13 и значениях $R1 = R2 = R = 10$ кОм, $R_k = 1$ кОм, $E = 5$ В, $U^0 = 0,2$ В, $U^1 = 2,4$ В, $\beta = 50$, $C_{ш} = 200$ пФ определите t_{Φ}^+ , t_{Φ}^- , t_{Φ}^+ / t_{Φ}^- , считая транзистор безынерционным.

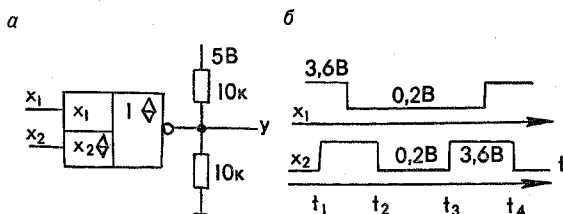


Рис. 4.82

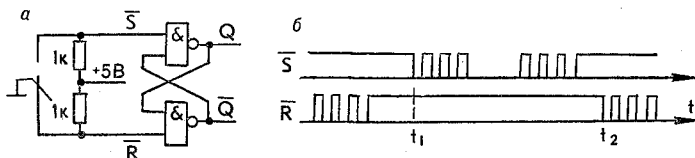


Рис. 4.83

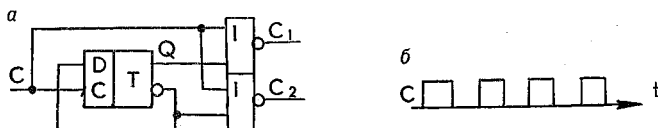


Рис. 4.84

4.5. Определите $U_{п}^{\pm}$ для схемы ДТЛШ (рис. 4.15) при $U_{вх}^1 = 2,7$ В, $U_{вх}^0 = 0,5$ В, $U_{д.ш.0} = 0,4$ В, $U_{д.ш.3} = 0,3$ В, $U_{э.3} = 0,6$ В, $U_{э.0} = 0,7$ В.

4.6. Для элемента с тремя состояниями выхода (рис. 4.82, а) изобразите изменение напряжения на выходе y при заданных входных сигналах (рис. 4.82, б). Схема приведена на рис. 4.18.

4.7. На рис. 4.83, а показана схема защиты от колебаний механического ключа, а на рис. 4.83, б — временная диаграмма на входах \bar{R} , \bar{S} триггера (ТТЛ серии 155) при переключении ключа. Изобразите изменения выходного сигнала Q .

4.8. Для схемы на рис. 4.84, а при входных тактовых импульсах (рис. 4.84, б) изобразите сигналы C_1 , C_2 . Исходное состояние триггера $Q = 0$.

4.9. На базе дешифраторов (см. рис. 4.46, в) и инвертора постройте трехразрядный двоичный дешифратор.

4.10. Напишите булеву функцию и приведите функциональную схему мультиплексора 2—1.

4.11. Реализуйте на MUX 4—1 (см. рис. 4.50) функцию $\overline{z_2 z_1}$.

4.12. Синтезируйте синхронный 2-разрядный суммирующий счетчик на D-триггерах с динамическим синхронизирующим входом.

Глава 5. ОРГАНИЗАЦИЯ МИКРОПРОЦЕССОРОВ И МИКРОЭВМ

5.1. Понятие об архитектуре микропроцессора и микроЭВМ

Базовым элементом микроЭВМ является микропроцессор (МП) — программно-управляемое устройство, предназначенное для обработки цифровой информации и управления процессом этой обработки. Микропроцессоры выполняются в виде одной или нескольких БИС.

Появление микропроцессоров существенно изменило процесс проектирования средств вычислительной техники. Разработчик системы, использующей микропроцессоры и микроЭВМ, проектирует систему на уровне связей между БИС, функциональными частями системы и устройствами. Разработчику недоступен уровень отдельных транзисторов, составляющих микропроцессор или микроЭВМ. Микропроцессор или микроЭВМ воспринимается разработчиком как нечто целое, имеющее свои внешние характеристики, заложенные в его архитектуре.

Архитектура микропроцессора или микроЭВМ — это аппаратные, микропрограммные и программные средства, которые создают организованную вычислительную среду, необходимую для обработки данных в соответствии с назначением микропроцессора или микроЭВМ.

Изучение архитектуры микропроцессора обычно начинают со знакомства с технологией изготовления МП, определяющей электрические (уровни логических сигналов, потребляемая мощность и т. п.) и динамические (максимальная тактовая частота, время задержки распространения) характеристики, функциональной ориентацией микропроцессора, его структурой, интерфейсом, системой команд или микрокоманд, возможностями расширения функций и совместимости с другими типами МП. Эти же вопросы служат классификационными признаками при анализе микропроцессоров. Рассмотрим один из вариантов классификации, когда классификационным признаком является число БИС МП. Различают *однокристалльный* (ОМП) и *многокристалльный* (ММП) микропроцессоры. Если весь процессор разме-

щен в одной БИС (СБИС), то его называют однокристалльным микропроцессором. При создании процессора более быстродействующего или большей разрядности или с расширенным набором команд его реализуют на нескольких БИС — это уже многокристальный МП.

Как правило, ОМП являются более простыми в применении, однако системы на их основе имеют меньшее быстродействие и менее приспособлены к решению задачи. ММП позволяют применять менее «плотные», но более быстродействующие технологии (ТТЛШ, ЭСЛ), имеют наращиваемую разрядность и являются микропрограммируемыми. Это позволяет создавать вычислительные устройства с любым набором команд, а при необходимости — с переменным набором команд. Недостатками ММП являются сложность разработки, а также большие габариты систем на их основе. Многокристальный микропроцессор может быть реализован на МПК с функциональным разделением на БИС (горизонтальная разрезка, например, УУ отдельно от ОУ). Такой МПК называют *модульным*. Если же разделение на отдельные БИС предполагает создание звеньев малой разрядности с возможностью их соединения между собой (вертикальная разрезка), то в этом случае МПК называют *разрядно-модульным* или *секционированным*, а отдельные звенья — *микропроцессорными секциями* (МПС).

В настоящее время в нашей стране выпускаются все указанные выше типы устройств: однокристалльные микроЭВМ (К1816, К1820), однокристалльные МП (К580, К1801), модульные МПК (К1810), секционированные МПК (К589, К1804, К1802, К1800).

5.2. Однокристалльные микропроцессоры

Общие сведения. Однокристалльный МП — это сложное устройство, состоящее из сотен тысяч элементов, которые связаны с внешним миром с помощью внешних выводов БИС (24, 40, 48 или 64 контакта). По внешним выводам в ОМП поступает энергия, необходимая для его работы, а также управляющие сигналы, входная информация для обработки. Кроме того, через эти выводы ОМП сообщает о своем состоянии, выдает результаты вычислений и служебную информацию.

Любой однокристалльный МП (рис. 5.1) должен иметь выводы питания (S — напряжение питания, G — общий), синхронизации (CLK), вход DI и выход DO информа-

ции, выход А адреса памяти (ведь память расположена вне МП, на других БИС), входы управления С, выходы состояния МП (ST).

Группу контактов МП, выполняющих общую функцию, называют шиной и иногда изображают на рисунках двумя параллельными линиями, объединенными общей стрелкой. Например, шины DI, DO и А содержат 8, 16

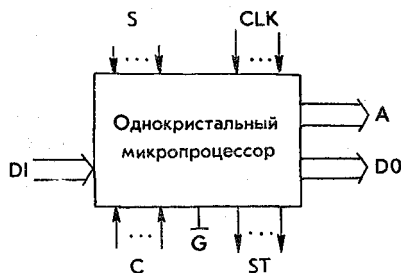


Рис. 5.1

или 32 контакта, шина С представляет собой совокупность сигналов СБРОС, ОСТАНОВ, ПУСК и других, в шину ST объединяются сигналы РАБОТА, ОЖИДАНИЕ, ОШИБКА, ПЕРЕПОЛНЕНИЕ и др. Очевидно, что чем большая разрядность шин, тем больше информации может принять, передать и обработать МП в единицу времени. Но большое число выводов усложняет технологию МП, снижает его надежность, увеличивает габаритные размеры. Поэтому разработчики МП стремятся сократить число контактов БИС МП.

Для этой цели широко применяется мультиплексирование. Под мультиплексированием (от англ. multiplex — сложный, многократный) здесь понимается совмещение функций некоторых шин.

Наиболее часто мультиплексированию подвергаются входная и выходная информация, которая поступает в МП и выдается из него по двунаправленной шине данных в разные интервалы времени. ОМП с двунаправленной магистралью D показан на рис. 5.2. Двунаправленная шина данных иногда обозначается DIO (от англ. data input-output — входные-выходные данные).

Следующий уровень мультиплексирования предполагает совмещение шин адреса и данных в одну двунаправленную магистраль адреса — данных AD (рис. 5.3). Такое мультиплексирование применяется в 16-разрядных

МП К1801ВМ1, К1801ВМ2, К1801ВМ3, К1810ВМ86. Магистраль AD имеет уже тройное назначение: передача адреса из МП, данных из МП и данных в МП. Двухнаправленные магистрали D и AD имеют возможность отключения, т. е. могут переводиться в высокоимпедансное состояние («третье состояние»), не нагружающее линию.

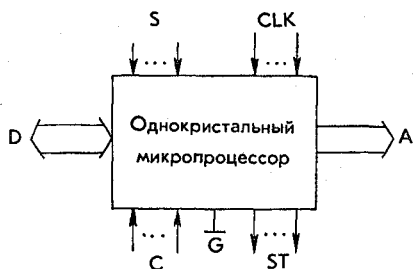


Рис. 5.2

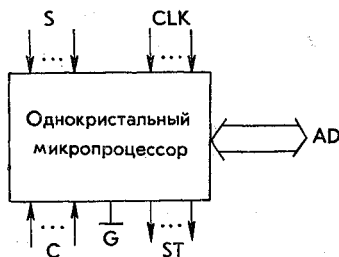


Рис. 5.3

В случае необходимости прибегают и к другим видам мультиплексирования, например совмещают часть информации о состоянии, т. е. шину состояния ST, с шиной данных. Этот принцип используется в МП КР580ВМ80. По шине данных в определенных тактах работы МП выдается «байт состояния». Иногда в ОМП совмещают отдельные выводы шин C и ST (управления и состояния).

Мультиплексирование, дающее возможность сокращать количество внешних выводов, приводит, однако, к потере быстродействия и некоторому усложнению схем.

Однокристалльный МП КР580ВМ80. На рис. 5.4 представлена структурная схема БИС КР580ВМ80 — широко распространенного однокристалльного микропроцессора. МП имеет внутреннюю двухнаправленную магистраль, развязанную от внешней магистрали данных D двухнаправленным буфером с тремя состояниями. Обработку информации выполняет арифметико-логическое устройство (АЛУ) со вспомогательными элементами: регистром-аккумулятором А, буферными регистрами БР1 и БР2, регистром признаков РП и схемой десятичной коррекции. Важное значение имеет блок регистров общего назначения (РОН), в который входят 8-разрядные регистры W, Z, B, C, D, E, H, L, 16-разрядный указатель стека (УС) и программный счетчик (ПС), а также регистр адреса (РА) и схема приращения (СхП). Работа с ре-

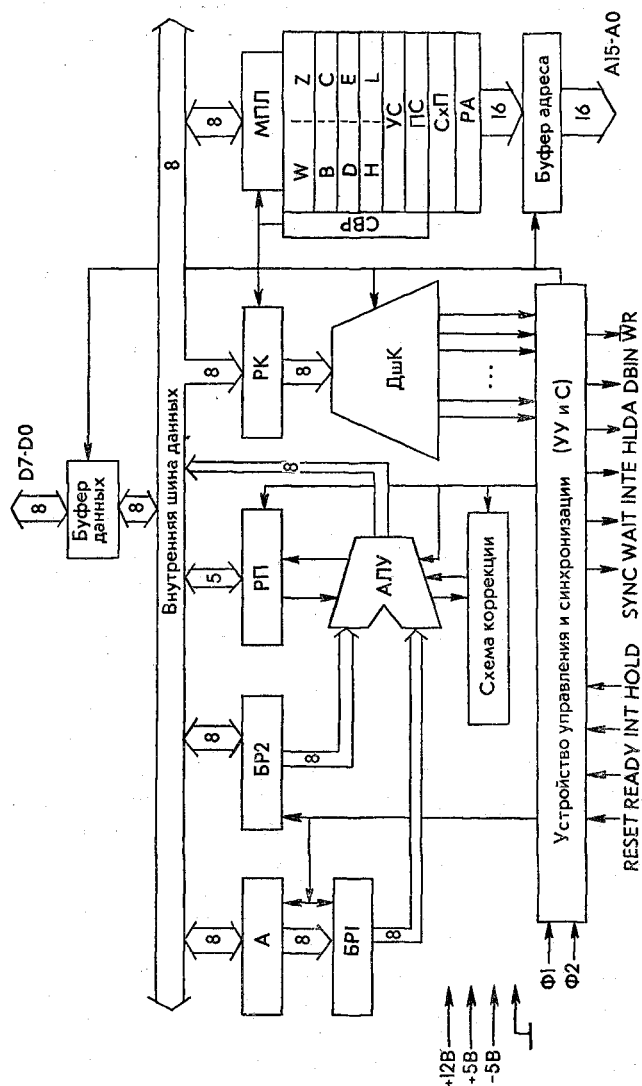


Рис. 5.4

гистрами поддерживается мультиплексором (МПЛ), схемой выбора регистра (СВР) и буфером адреса, выходящим на 16-разрядную шину адреса А и имеющим три состояния. Взаимодействие всех узлов микропроцессора организует устройство управления и синхронизации (УУиС).

МП работает под управлением программы, которая состоит из отдельных команд. 8-разрядный код команды поступает на регистр команд (РК), дешифрируется дешифратором команд (ДшК) и определяет работу устройства управления и синхронизации, которое синхронизирует взаимодействие всех остальных узлов микропроцессора. На УУиС поступают 4 внешних управляющих сигнала: RESET (СБРОС, начальная установка), READY (ГОТОВНОСТЬ памяти или порта ввода-вывода к обмену информацией с МП), INT (запрос на прерывание) и HOLD (запрос на захват шины). Входы $\Phi 1$ и $\Phi 2$ — это входы импульсов синхронизации — двух фаз. Из устройства управления на внешние выходы подается 6 сигналов: SYNC (СИНХР, определяющий, что на шину D МП выдал байт состояния), WAIT (ОЖИДАНИЕ, означающий, что МП находится в состоянии ожидания сигнала READY), INTE (разрешение прерываний), HLDA (подтверждение захвата), DBIN (ЧТЕНИЕ, означающий, что буфер данных включен на прием информации с шины D в МП), \overline{WR} (ЗАПИСЬ, указывающий, что МП выдает информацию на шину D).

МП КР580ВМ80 расположен в 40-выводном корпусе и требует подключения трех питающих напряжений: +12 В, +5 В, —5 В. Вывод «Общий» подключается в общей точке «0В» источников питания. Как только на выходы $\Phi 1$ и $\Phi 2$ подается двухфазная последовательность тактовых импульсов (рис. 5.5) с частотой 2—2.5 МГц, микропроцессор начинает функционировать.

Функционирование микропроцессора заключается в постоянном выполнении команд. Выполнив одну команду, МП выбирает из памяти вторую, выполнив вторую, выбирает третью, выполняет ее и т. д. Таким образом, обработка каждой команды состоит из двух этапов: *этапа выборки* и *этапа исполнения* (рис. 5.6). Нарушение этого процесса происходит лишь в случае прерывания, а также в состояниях ожидания, останова и захвата, в которых может находиться МП. Команды программы находятся в памяти, поступают в МП по шине данных и запоминаются в регистре команд.

Выполнение любой команды в МП КР580ВМ80 состоит из циклов. В зависимости от сложности команды для ее выполнения требуется от 1 до 5 циклов: М1 — М5. Первый цикл М1 для любой команды соответствует этапу выборки и называется *цикл выборки*. Каждый цикл длится от 3 до 5 тактов, обозначаемых $T1$ — $T5$. Такты $T1$, $T2$, $T3$ присутствуют в каждом цикле, такты $T4$, $T5$ — в некоторых циклах.

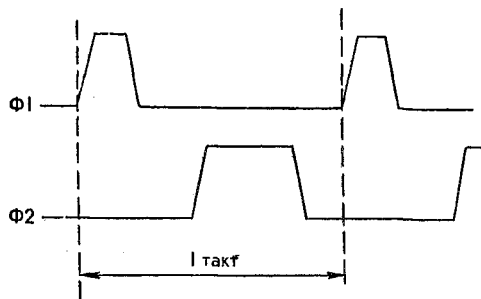


Рис. 5.5

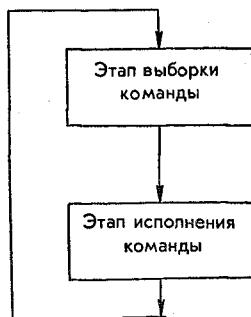


Рис. 5.6

В такте $T1$ МП выдает на шину А адрес памяти, а на шину D — байт состояния, сопровождаемый сигналом SYNC. Байт состояния представляет собой набор сигналов, характеризующих состояние МП в текущем цикле и совместно с сигналами DBIN, \overline{WR} используется для управления блоками микроЭВМ.

Во втором такте $T2$ МП анализирует вход READY. Если вход в пассивном состоянии (напряжение высокого уровня), то МП перейдет в дополнительный такт T_w , который будет продолжаться до появления активного уровня READY, после чего МП переходит к такту $T3$, в котором выполняется обмен по шине D. В *цикле чтения* происходит передача информации в процессор, которая сопровождается сигналом DBIN (л). В *цикле записи* данные выдаются из МП и сопровождаются сигналом \overline{WR} (ц). На рис. 5.7 и 5.8 приведены временные диаграммы работы МП в циклах чтения и записи соответственно. Цикл чтения показан с появлением и ожиданием готовности, цикл записи — для случая обмена с быстродействующей памятью, когда ожидание готовности T_w отсутствует.

Программно-доступные средства ОМП. К программно-доступным средствам ОМП КР580ВМ80 относятся

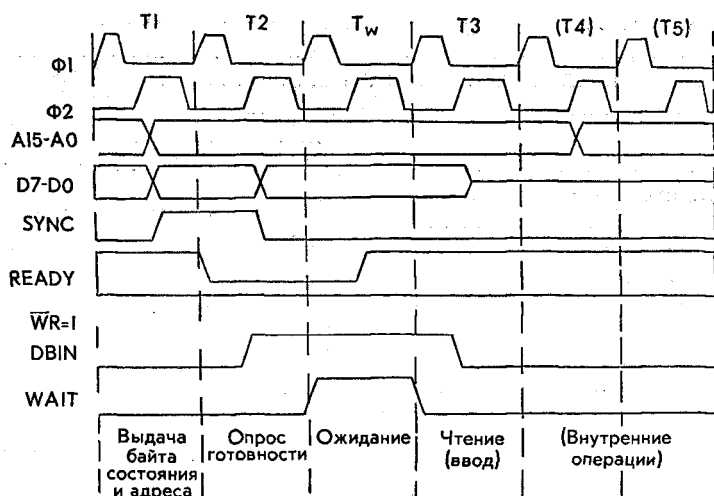


Рис. 5.7

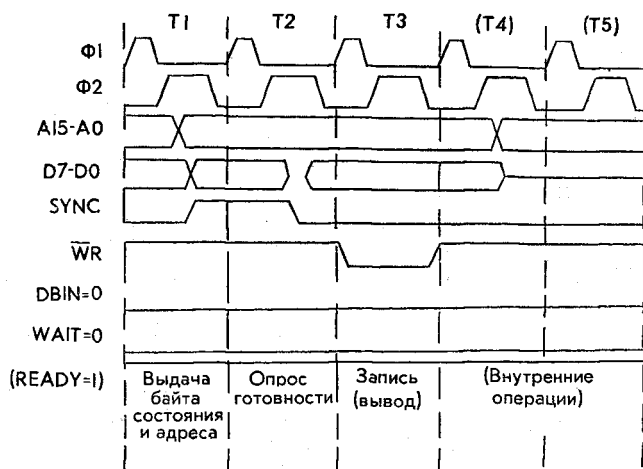


Рис. 5.8

внутренние 8-разрядные регистры МП: А, В, С, D, Е, Н, L. *Регистр-аккумулятор* А играет особую роль: в большинстве команд, реализующих двухместные операции, один из операндов находится в аккумуляторе; результат также всегда помещается в аккумулятор. *Регистры*

В, С, D, E, H, L (РОН) используются для хранения промежуточных результатов, констант, адресов данных и т. п. Они могут также использоваться как 16-разрядные регистровые пары: В, С — пара В; D, E — пара D; H, L — пара H. Пара H, L обычно используется для хранения адреса памяти, в этом случае в команде нет необходимости явно указывать адрес данных, а пара H, L называется *регистром косвенного адреса*. Содержимое любого регистра можно переслать в другой, записать в память, загрузить из памяти, использовать в качестве операнда в арифметических и логических операциях.

К программно-доступным средствам относится также 16-разрядный *программный счетчик* (ПС) — регистр, содержащий адрес следующей команды, которая будет выбрана из памяти после выполнения текущей. Программный счетчик обозначают иногда РС (от англ. program counter). Такое название он получил потому, что при выборке каждого очередного байта команды его содержимое автоматически увеличивается на единицу, обеспечивая тем самым доступ к следующему байту. Кроме того, специальными командами его содержимое может изменяться. Эти команды называют командами переходов или передачи управления.

В МП КР580ВМ80 имеется еще один специальный 16-разрядный регистр — *указатель стека* УС (или SP от англ. stack pointer). В этом регистре хранится адрес вершины стека — специальной области памяти, используемой для временного хранения данных и адресов.

При выполнении арифметических и логических операций в МП формируются признаки результата, которые помещаются в программно-доступный *регистр признаков* РП, обозначаемый иногда F (от англ. flags — флажки). В этот регистр заносятся 5 признаков результата: S — знак результата («1» — минус, «0» — плюс), Z — признак нулевого результата («1» — результат нулевой, «0» — результат ненулевой), C — перенос из старшего разряда («1» — есть перенос, «0» — нет переноса), P — паритет, четность («1» — в результате четное число единиц, «0» — нечетное), AC — дополнительный перенос — из младшей тетрады в старшую (используется при обработке чисел, представленных в двоично-десятичном коде). Расположение признаков в разрядах РП показано на рис. 5.9.

К программно-доступным средствам, находящимся

внутри МП, относится *триггер разрешения прерываний*, расположенный в устройстве управления, который может устанавливаться и сбрасываться специальными командами.

Построение микроЭВМ. МикроЭВМ на базе ОМП включает центральный процессор, память (ОЗУ и ПЗУ) и внешние устройства (ВУ — клавиатуру, дисплей, печат-

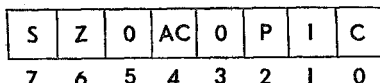


Рис. 5.9

тающее устройство и т. п.), подключаемые к шинам МП через специальные схемы — порты ввода-вывода. Синхронизацию процессора осуществляет тактовый генератор (для МП КР580ВМ80 можно использовать специально разработанную БИС КР580ГФ24). Для увеличения нагрузочной способности шин адреса включаются шинные формирователи. Буферизацию двунаправленной шины данных, запоминание байта состояния МП и формирование системных управляющих сигналов осуществляет БИС системного контроллера КР580ВК28. Дешифраторы Дш формируют сигналы включения соответствующего устройства или схемы, когда на шине адреса появляется его адрес.

На рис. 5.10 приведена структура микроЭВМ на базе ОМП КР580ВМ80.

Системный контроллер в каждом цикле вырабатывает один из пяти системных управляющих сигналов: \overline{MEMR} — чтение из памяти, \overline{MEMW} — запись в память, I/OR — ввод, I/OW — вывод, \overline{INTA} — подтверждение прерывания. Два первых сигнала управляют обменом между МП и памятью микроЭВМ, два следующих сигнала выполняют обмен между МП и портом ввода или портом вывода. Сигнал I/OR появляется при выполнении процессором команды IN с адресом порта ввода, I/OW — при выполнении команды OUT с адресом порта вывода. Порт выбирается своим дешифратором при появлении на шине адреса кода, присвоенного этому порту (от 0 до 255).

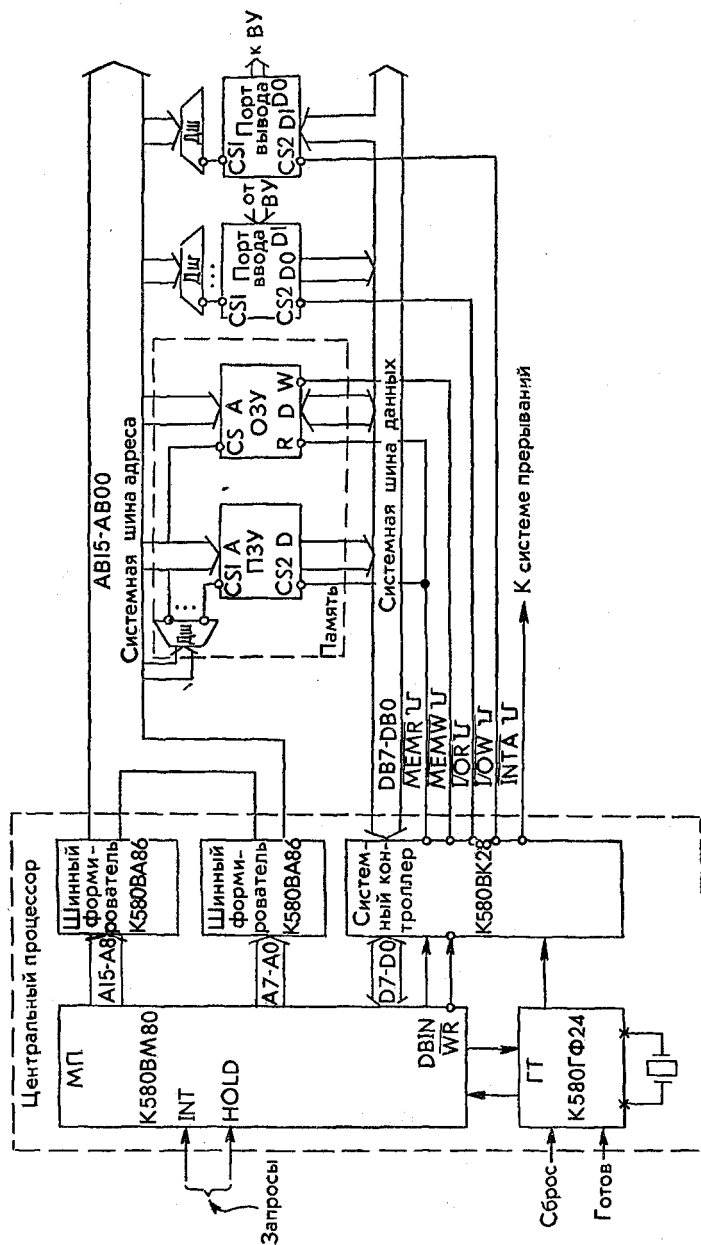


Рис. 5.10

5.3. Многокристальные микропроцессоры

Устройство многокристального микропроцессора (ММП). Если в однокристальном МП в одной БИС размещались и операционная (АЛУ, РОНЫ, буферы и т. д.) и управляющая части (регистр и дешифратор команд, устройство управления и синхронизации и др.), то в многокристальном эти две части размещены в разных БИС, точнее, в разных группах БИС. Управляющую часть называют *блоком микропрограммного управления* (БМУ). Операционная часть состоит из микропроцессорных секций, схем ускоренного переноса, регистров общего назначения и других. Нередко в состав МПК включают специализированные БИС для ускорения работы операционной части — арифметические расширители (умножители, сдвигатели и т. п.).

Микропроцессорный комплект БИС К1804. МПК БИС К1804 — секционированный, выполнен на основе технологии ТТЛШ. Он предназначен для построения контроллеров различной организации, микро- и мини-ЭВМ различного назначения с любой архитектурой и набором команд, а также измерительных систем, систем обработки данных. Его зарубежный аналог — ИМС семейства Аm2900. Состав МПК приведен в табл. 5.1.

БИС К1804ВС1. Структурная схема БИС К1804ВС1 — четырехразрядной микропроцессорной секции (МПС) — приведена на рис. 5.11. Микросхема содержит АЛУ, двухпортовое регистровое ЗУ (РЗУ), регистры РА, РВ, РQ, сдвигатели САЛУ и CQ, мультиплексоры МА, МВ, МУ, буфер Y, схему формирования признака нуля (СПН) и блок управления (БУ). АЛУ МПС выполняет восемь операций над данными — три арифметические (сложение, вычитание, обратное вычитание) и пять логических поразрядных (такие, как И, ИЛИ, исключающее ИЛИ и т. п.). Операция АЛУ задается кодом в разрядах I5, I4, I3 микроконструкции. Наличие мультиплексоров МА и МВ позволяет выбирать в качестве операндов АЛУ входную шину D, РА, РВ, РQ или 0. Одно из восьми сочетаний ($A - Q$, $A - B$, $0 - Q$, $0 - B$, $0 - A$, $D - A$, $D - Q$, $D - 0$) выбирается кодом I2, I1, I0. Наконец, сдвигатели САЛУ, CQ и мультиплексор МУ управляются разрядами I8, I7, I6, что позволяет заносить в РЗУ сдвинутый или несдвинутый результат операции АЛУ, при этом сдвигать или не сдвигать содержимое РQ, выдавать на шину Y либо результат АЛУ, либо содержимое РА

Таблица 5.1

Обозначение БИС	Функциональное назначение БИС
K1804BC1	Микропроцессорная секция (МПС)
K1804BC2	Микропроцессорная секция с расширенными возможностями (МПС)
K1804BP1	Схема ускоренного переноса (СУП)
K1804BP2	Схема управления состоянием и сдвигами (СУСС)
K1804BY1	Схема управления адресом микрокоманды (СУАМ)
K1804BY2	То же
K1804BY3	Схема управления следующим адресом (УСА)
K1804BY4	Схема управления последовательностью микрокоманд (УПМ)
K1804BY5	Секция управления адресом программной памяти (УАПМ)
K1804IP1	Четырехразрядный регистр
K1804IP2	Восьмиразрядный регистр
K1804IP3	Восьмиразрядный двунаправленный регистр
K1804BA1	Четырехразрядный канальный приемопередатчик
K1804BA2	То же
K1804BA3	Четырехразрядный канальный приемопередатчик с интерфейсной логикой
K1804BI1	Схема векторного приоритетного прерывания
K1804ГГ1	Системный тактовый генератор

(выходной буфер открывается сигналом \overline{OE}). Адрес $A_0 — A_3$ задает читаемый РОН по каналу А, адрес $B_0 — B_3$ — читаемый по каналу В и выбираемый для записи РОН. РЗУ может в одном такте быть источником и приемником результата, что обеспечивается записью в регистры РА, РВ и РЗУ в противофазе тактовых импульсов.

Сдвигатели САЛУ и СQ имеют двунаправленные выводы для старшего PR3 (PQ3) и младшего PR0 (PQ0) разрядов с третьим состоянием и работают по схеме, представленной на рис. 5.12.

Двунаправленные входы-выходы сдвигателей, вход С0 и выход С4 переноса предназначены для соединения МПС в многоразрядные МП. Один из вариантов соединения МПС в 12-разрядный микропроцессор приведен на рис. 5.13.

Управление на МПС ($I_8 — I_0$, $A_0 — A_3$, $B_0 — B_3$, OE) и синхронизация (CLK) подаются параллельно. Многоразрядные входная и выходная шины состояются из шин МПС. Последовательный перенос и входы-выходы сдви-

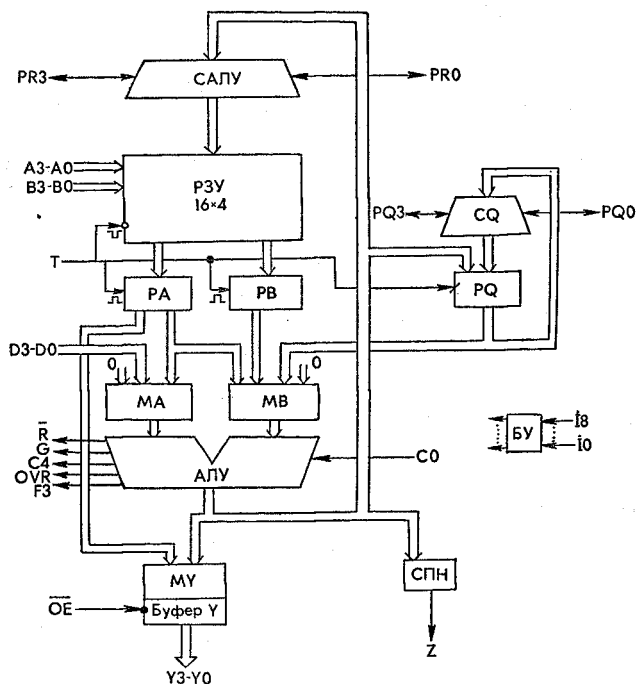


Рис. 5.11

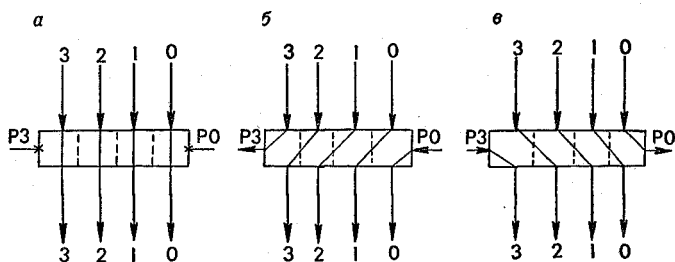


Рис. 5.12

гов соединяются последовательно. Включение мультиплексоров МUX с третьим состоянием на выходе позволяет реализовать богатый набор различных вариантов сдвигов. При сдвигах влево включаются мультиплексоры 3, 4, вправо — мультиплексоры 1, 2. Коммутируя линии $PR0 - PR3$ и $PQ0 - PQ3$, можно осуществлять независи-

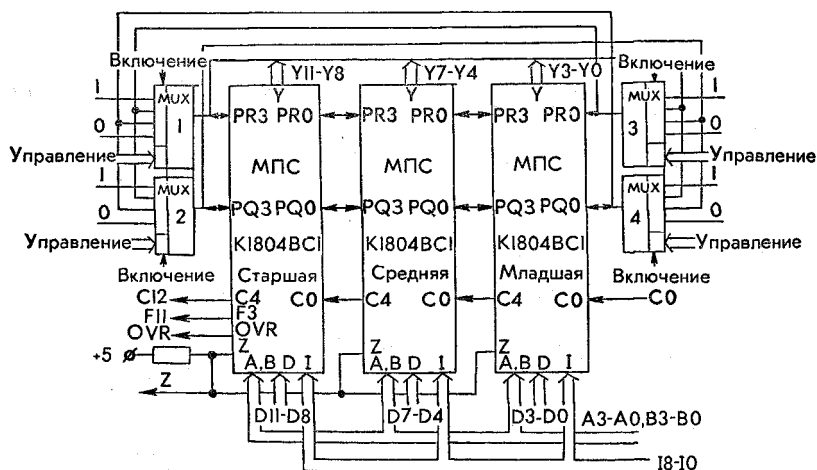


Рис. 5.13

мые сдвиги АЛУ и PQ. Коммутация PR—PQ и RQ—PR позволяет совместить сдвиг АЛУ и PQ как единого слова двойной длины. Наконец, коммутация констант 1 или 0 позволяет вводить их, при необходимости, в освобождающиеся разряды.

Управление БИС К1804ВС1 осуществляется 18 разрядами микрокоманды.

БИС К1804ВС2. На рис. 5.14 приведена структурная схема БИС К1804ВС2, имеющей более широкие функциональные возможности по сравнению с К1804ВС1. Вместо мультиплексора МУ и буфера У введены доступные снаружи буферы DY и DB. Корпус вместо 40 имеет 48 выводов. Совпадающие по названию узлы (сдвигатели, АЛУ) в БИС К1804ВС2 несколько усложнены. АЛУ выполняет 16 простых операций (7 арифметических, 7 логических, константа 0 и константа 1), а также 9 специальных функций, среди которых есть и многотактные операции (умножение, деление).

Источники операндов задаются сочетанием сигналов на управляющих входах \overline{EA} , \overline{OEB} и I_0 . Это могут быть PA, PB, DA, DB, PQ в следующих сочетаниях: PA—PB, PA—DB, PA—PQ ($\overline{EA}=0$), DA—PB, DA—DB, DA—PQ ($\overline{EA}=1$). В случаях PA—PQ и DA—PQ возможны варианты обмена по шине DB, работающей на выдачу ($\overline{OEB}=0$) либо выключенной ($\overline{OEB}=1$).

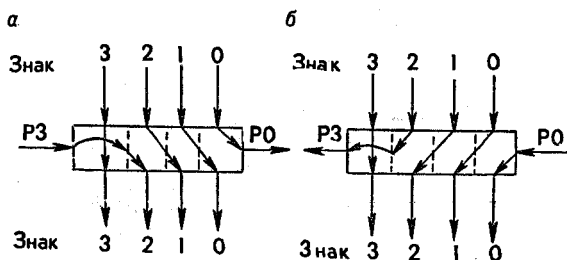


Рис. 5.15

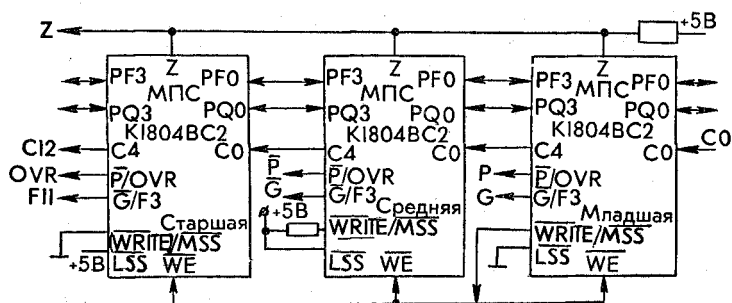


Рис. 5.16

МПС (при $LSS=0$) это выход WRITE для управления входом WE (разрешение записи в РЗУ) всех секций; при $LSS=1$ эта линия становится входом выбора старшей МПС ($MSS=0$) или средней МПС ($MSS=1$). Соединение трех МПС показано на рис. 5.16.

Кроме рассмотренных выше управляющих входов, имеются входы OEY (разрешение выдачи результата АЛУ на шину DY) и IEN (разрешение записи в PQ). Выход признака нуля Z в некоторых специальных функциях используется как дополнительный управляющий вход. Так как выходы АЛУ F3 (знак) и OVR (переполнение) используются только в старшей МПС, а выходы P и G для формирования ускоренного переноса — только в средних и младшей МПС, то эти четыре сигнала мультиплексируются в две линии G/F3 и P/OVR в зависимости от позиции МПС. Всего для управления секцией K1804BC2 требуется 22 разряда микрокоманды (не считая управления переносом и сдвигами).

БИС K1804BP2, K1804BP1. При построении ЭВМ на

базе МПС возникает необходимость введения некоторых внешних схем. На рис. 5.13 показан вариант организации широкого набора различных сдвигов в МПС К1804ВС1 с помощью внешних мультиплексоров. Подобной поддержки требует управление входным переносом в младшую МПС. Необходимо также отметить, что признаки результата, вырабатываемые после каждой операции АЛУ, являются по существу сигналами, возникшими на выходе комбинационной схемы. В следующем такте, после смены операндов и (или) микроинструкции, они «исчезнут», т. е. сменятся новыми. Здесь речь идет о признаках С4 (выходной перенос), OVR (переполнение), F3 (знак), Z (признак нулевого результата). Для того чтобы их можно было использовать, признаки должны запоминаться в регистре. Такой регистр называют регистром состояния. Анализ различных сочетаний признаков позволяет определить выполнение или невыполнение какого-либо условия ($A > B$, $A \geq B$, $A = B$, $A < B$, $A \leq B$, $A \neq B$ и др.).

Достаточно сложная логика управления сдвигами, запоминания состояния, управления переносом и проверки различных условий собрана воедино в БИС К1804ВР2 — в *схеме управления состоянием и сдвигами (СУСС)*. Данная схема состоит из нескольких практически не связанных между собой частей: блока управления сдвигами, блока управления переносом, блока обработки признаков и блока проверки условия. Все блоки работают как комбинационные схемы, за исключением двух 4-разрядных регистров состояния, которые принимают информацию по переходу тактового сигнала из 0 в 1 (\uparrow). Хранимые в них признаки обозначаются буквами: N (знак), Z (ноль), V (переполнение) и C (перенос).

БИС имеет 13 входов микроинструкции, определяющей выполняемую операцию. Это входы I0 — I12. Поле микроинструкции разбито на группы: I0 — I5 — управление регистрами состояния, в том числе I0 — I3 — управление проверкой условия. Разряды I6 — I10 управляют сдвигами (32 варианта сдвигов), а I11, I12 — управляют переносом. Кроме шины микроинструкции, БИС К1804ВР2 имеет другие входы управления: EC, EN, EV, EZ — входы разрешения записи признаков; CEM — вход разрешения записи в RгM; CEN — вход разрешения записи в Rм; OEU — выдача информации; OECT — вход кода условия; SE — вход разрешения сдвига.

Таким образом, при микропрограммировании БИС К1804BP2 требует 22 управляющих разряда. Пример включения БИС К1804BP2 совместно с БИС МПС К1804BC1 показан на рис. 5.17. Подобным способом БИС СУСС соединяется с МПС К1804BC2.

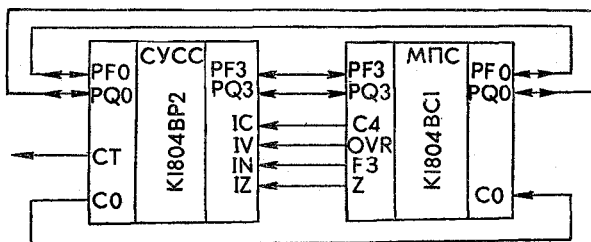


Рис. 5.17

Как и большинство секционированных МПК БИС, МПК К1804 имеет в своем составе ИС СУП (схему ускоренного переноса) К1804BP1, которая, будучи подключенной к четырем МПС К1804BC1 или К1804BC2, реализует одновременный перенос во все три старшие секции. ИС К1804BP1 имеет стандартную организацию СУП и совпадает по функции и расположению выводов с такими ИС СУП, как К155ИП4.

5.4. Микропрограммное управление

Каждое действие в микроЭВМ выполняется под воздействием сигналов управления на все БИС операционной части, внешние мультиплексоры, буферы, триггеры, регистры, память. Совокупность таких управляющих сигналов в текущем такте работы микроЭВМ называют микрокомандой (МК). К микрокомандам относят также и то действие (преобразование информации в МП системе), к которому приводит данная совокупность управляющих сигналов. Микрокоманда как действие состоит из элементарных событий, называемых микрооперациями. Микрооперация — это преобразование информации в каком-либо функциональном узле. Например, микрокоманда сложения двух операндов А и В и занесения сдвинутого влево результата на место операнда В состоит из элементарных микроопераций: 1) подача А и В на входы АЛУ; 2) выполнение в АЛУ сложения; 3) сдвиг результата в сдвигателе; 4) запись результата

из сдвигателя по указанному адресу. Все эти микрооперации выполняются в МПС за один такт.

Микропрограммное управление операционными БИС — это, как правило, «многоразрядное» управление. Как уже указывалось, БИС МПС К1804ВС1 требует для управления 18 разрядов, К1804ВС2 и К1804ВР2 — 22 разряда. Таким образом, для управления только двумя БИС — МПС и СУСС — в микрокоманде должно быть более 40 разрядов, а при необходимости задания констант — до 60 разрядов.

Сложное преобразование информации в операционной части МП системы, требующее нескольких тактов для выполнения, реализуется некоторой последовательностью микрокоманд, называемой микропрограммой. Так, микропрограммами являются умножение чисел в формате с плавающей запятой, извлечение корня квадратного и др.

Микропрограммы различных операций хранятся в микропрограммной памяти (МПП). Для выполнения каждой микрокоманды необходимо задать ее адрес и прочитать из МПП. Если микропрограмма является *линейной последовательностью микрокоманд*, то задавать адрес каждой следующей МК можно счетчиком, предварительно загрузив в него начальный адрес микропрограммы. Однако линейные микропрограммы встречаются редко. Чаще они представляют собой *последовательности с ветвлениями* (т. е. выбором одной из двух и более МК в качестве следующей в зависимости от неизвестных заранее признаков или условий) и *с циклами* (т. е. с многократным повторением некоторой группы микрокоманд и выходом из цикла по счетчику повторений или по другому условию). В таком случае адрес следующей микрокоманды должен определяться специальным образом. Определение адреса следующей МК и есть основное назначение БИС БМУ. Исходными данными для этого служат обычно текущий адрес МК, специальные разряды в поле текущей МК, а также совокупность признаков и условий, поступающих от операционной части и из внешней аппаратуры. Обобщенная структура БМУ показана на рис. 5.18.

БИС К1804ВУ1 и К1804ВУ2. Секции управления адресом микрокоманды (СУАМ) К1804ВУ1 и К1804ВУ2 размещены в 28-выводном и 20-выводном корпусах соответственно. Обе секции имеют разрядность, равную 4, и возможность ее наращивать.

Принцип работы СУАМ несложен: она позволяет выбрать в качестве адреса следующей МК содержимое одного из трех внутренних источников — РАМК (регистра адреса МК), СМК (счетчика микрокоманд), стека (глубиной 4 ячейки) или состояние внешних входных шин D3 — D0. Кроме того, имеется возможность отклю-

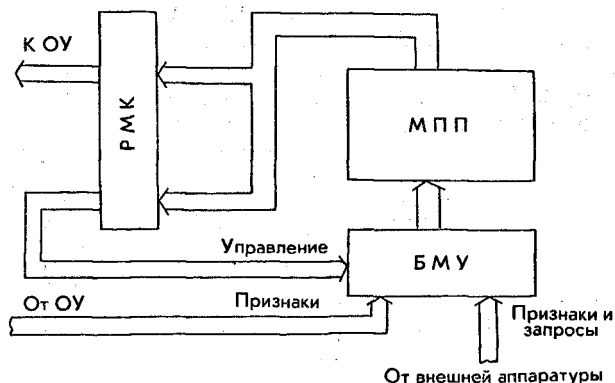


Рис. 5.18

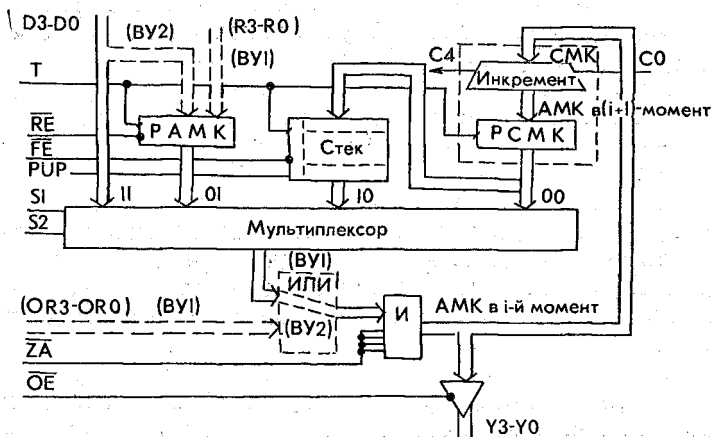


Рис. 5.19

чения выходного буфера (вход $\overline{OE} = 1$), выдачи нулевого адреса (вход $\overline{ZA} = 0$) и маскирования произвольных разрядов адреса МК, т. е. установки их в 1 (входы $\overline{OR3} - \overline{OR0}$ для $BY1$). Структурная схема СУАМ приведена на рис. 5.19.

РАМК позволяет по сигналу $\overline{RE} = 0$ загрузить произвольный адрес МК и в любом такте затем использовать его в качестве адреса следующей МК. Инкрементор СМК является комбинационной схемой и в то время как на $Y_3 - Y_0$ выдается адрес i -й МК, в СМК заносится код $i + 1$. Более сложная логика работы у стека. Назначение стека — хранить адрес возврата при переходе на подпрограмму либо адрес начала цикла при организации циклов. Так, при переходе из i -й МК к следующей микропрограмме по сигналу $PUP = 0$ в стек «заталкивается» код $i + 1$ из СМК, т. е. адрес, к которому нужно вернуться после выполнения микропрограммы. При этом должно быть $\overline{FE} = 0$. Содержимое указателя стека (адрес ячейки) уменьшается. Возврат из микропрограммы должен обеспечить $\overline{FE} = 0$ и $PUP = 1$ (чтение из стека). Код $i + 1$ читается на выходные шины $Y_3 - Y_0$, а указатель стека увеличивает свое содержимое на единицу, т. е. использованный адрес возврата «выталкивается» из стека. Глубина стека, равная четырем, позволяет выполнять соответствующее количество вложений микропрограмм.

БИС K1804ВУЗ. Для управления адресом следующей МК на управляющие входы СУАМ необходимо подавать соответствующие комбинации сигналов. Из рис. 5.19 видно, что СУАМ имеет 7 управляющих входов: S_1 , S_0 , \overline{FE} , PUP , \overline{ZA} , OE и \overline{RE} . Таким образом, для перехода к микропрограмме необходимо подать на указанные входы комбинацию 110110X, если адрес микропрограммы — на шине (или комбинацию 010110X, если адрес микропрограммы — в РАМК).

Все возможные условия ветвления можно свести к проверке одного разряда, отображающего выполнение условия. Так, например, для перехода по условию $ЕСЛИ \geq 0$ рассматривается знак результата (S). Если знак — минус ($S = 1$), то условие не выполняется, и на выход разряда условия нужно подать 0. При знаке результата плюс ($S = 0$) условие $ЕСЛИ \geq 0$ выполняется и на выход разряда условия подается 1. При этом микрокоманды, соответствующие условиям $S = 1$ и $S = 0$, располагаются в соседних ячейках МПП, и условие S непосредственно участвует в формировании адреса следующей микрокоманды в качестве младшего разряда.

Схема управления следующим адресом (УСА) K1804ВУЗ реализует изложенные выше функции. Схема содержит ПЗУ 32×8 , имеющие 5 адресных входов: I_3 , I_2 , I_1 , I_0 и TST, и 8 выходов: S_1 , S_0 , \overline{FE} , PUP — для

управления СУАМ, \overline{CTL} и \overline{STE} — для управления счетчиком циклов, \overline{PE} и \overline{ME} — для управления другими узлами. Кроме того, K1804BY3 имеет вход \overline{OE} (разрешение выдачи). Схема УСА помещена в 16-выводной DIP-корпус, расположение выводов ее совпадает с микросхемой K155PE3 (ППЗУ 32×8). Четыре входа I3 — I0 позволяют закодировать 16 различных инструкций, таких, как JZ — переход по нулевому адресу ($AMK = 0$), CJS — условный переход к подпрограмме: AMK выбирается либо из CMK (нет перехода), либо с шины (есть переход, при этом выполняется запись в стек), и другие.

На пятый вход TST подается значение рассматриваемого условия. Для того чтобы можно было анализировать различные условия, их подключают через мультиплексор кодов условий (MKY). Поскольку каждому условию может быть поставлено в соответствие противоположное (например, ЕСЛИ $= 0$ и ЕСЛИ $\neq 0$), то предусматривается возможность выдачи прямого либо инвертированного кода условия. На рис. 5.20 приведена структура БМУ на базе СУАМ K1804BY2 и УСА K1804BY3.

БИС K1804BY4. Объединение трех секций СУАМ, УСА и счетчика циклов в одну БИС привело к появлению БИС управления последовательностью микрокоманд (УПМ) — K1804BY4. БИС УПМ имеет 12-разрядную шину, что позволяет формировать адрес МПП емкостью до 4096 микрокоманд, и выполняет 16 микроинструкций, организующих циклы, переходы, обращения к подпрограммам и возвраты. Набор инструкции для УПМ почти совпадает с набором для УСА.

Для построения законченного БМУ в дополнение к БИС УПМ K1804BY4 требуются мультиплексор условий, преобразователь кода команды в начальный адрес микропрограммы (ПКК), микропрограммная память и регистр микрокоманды.

Рассчитаем ориентировочное количество разрядов микрокоманды. Для управления МПП емкостью 4096 ячеек поле перехода AMK составляет 12 разрядов. Для управления БИС УСА или УПМ требуются еще 4 разряда, 4 разряда необходимы для управления мультиплексором условий, позволяющим выбирать одно условие из шестнадцати, 1 разряд — для инвертирования кода условия. Таким образом, минимальное число разрядов адресной части микрокоманды равно 2. Общая же длина микрокоманды, включая и управление операционными

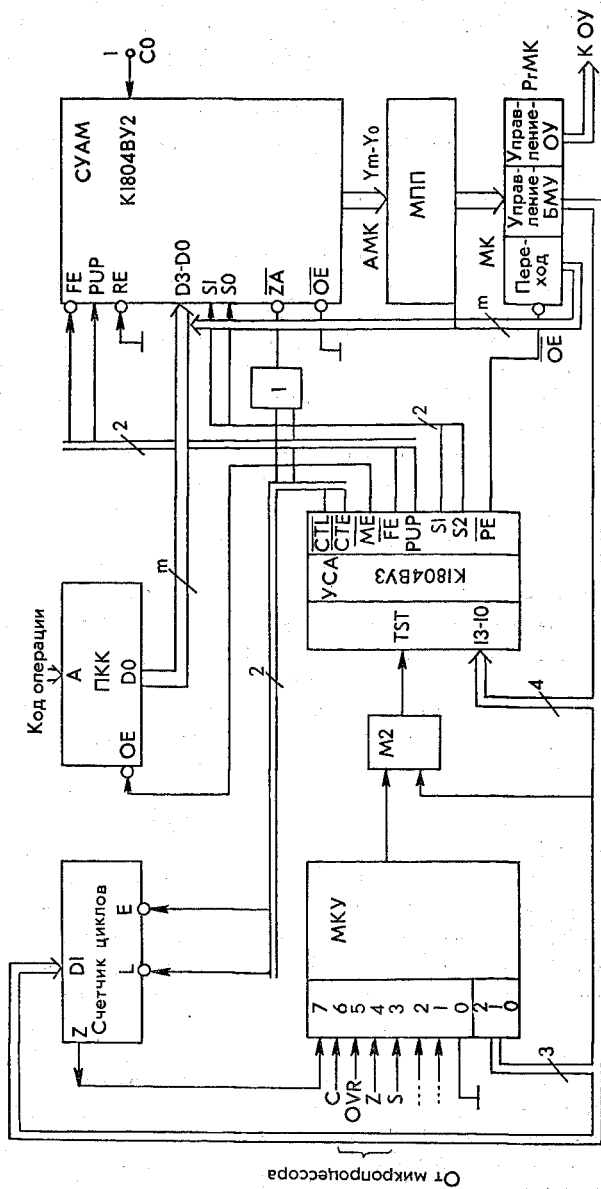


Рис. 5.20

БИС, составит около 80 разрядов, а с учетом ряда дополнительных компонентов микроЭВМ (умножителей, сдвигателей, преобразователей информации) формат микрокоманды может составлять 100—120, иногда 160—200 разрядов.

5.5. Организация памяти микроЭВМ

Иерархическая организация памяти. Одной из основных функциональных частей микроЭВМ является память — совокупность аппаратных и программных средств, предназначенных для записи, хранения и выдачи информации.

Память ЭВМ имеет иерархическую (многоуровневую) структуру. Более низкий уровень иерархии характеризуется меньшим быстродействием и большим объемом. В больших и суперЭВМ число уровней памяти достигает 5—6. В микроЭВМ принято выделять три уровня: сверхоперативную (СОЗУ), оперативную (ОЗУ) и внешнюю (ВЗУ) память.

Сверхоперативная память. СОЗУ имеет самые низкие время доступа (для БИС ТТЛ 50—100 нс) и объем (8—32 ячейки) и используется для временного хранения команд и данных, как правило, в течение выполнения одной или нескольких операций.

Функции сверхоперативной памяти часто выполняют регистры центрального процессора.

Оперативная память. ОЗУ характеризуется тем, что время поиска ячейки, в которой хранится информация, не зависит от ее адреса (расположения в памяти), а разрядность обычно равна разрядности шины данных и кратна восьми.

В ЭВМ адреса ячеек принято нумеровать целыми положительными числами 0, 1, 2, ..., $N - 1$, где N — общее число ячеек памяти, связанное с разрядностью g шины адреса формулой $N = 2^g$. Совокупность возможных разных адресов образует адресное пространство.

В каждой конкретной микроЭВМ не обязательно, чтобы все адресное пространство было заполнено физически подключенной памятью. На рис. 5.21 показана ситуация, когда физическая память занимает четверть адресного пространства (адреса указаны 16-ричной системой счисления). Как видно из рисунка, физическая память может быть расположена в разных четвертях адрес-

ного пространства, адреса физической памяти при этом будут различными, а объем один и тот же: 16 К байт. При разбиении адресного пространства на 4, 8, 16 частей говорят о разделении памяти на банки. В зависимости от расположения физический банк памяти имеет свой номер. Чтобы его дешифровать в адресном простран-

0000 3FFF	0-й банк
4000 7FFF	1-й банк
8000 BFFF	2-й банк
C000 FFFF	3-й банк

Рис. 5.21

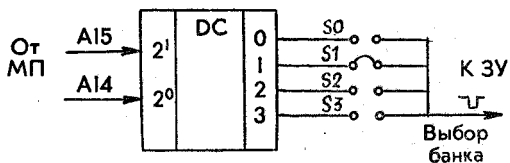


Рис. 5.22

стве, разделенном на четыре части, два старших разряда A15, A14 адресной шины подают на входы дешифратора DC (рис. 5.22).

Перемычками S0 — S3 задается номер банка, т. е. положение физической памяти 16 К байт в адресном пространстве 64 К. Перемычка должна быть установлена лишь одна. Для схемы на рис. 5.22 ЗУ 16 К байт включено в 1-й банк, следовательно, ему присвоены адреса с 4000_{16} по $7FFF_{16}$ включительно.

Постоянная память (ПЗУ). Помимо оперативной памяти, микроЭВМ обычно имеют и постоянную память, в которой хранятся системные программы, тесты, трансляторы, прикладные программы. Информация, записанная в ПЗУ, не подлежит изменению и предназначена только для чтения. Отсюда иногда встречающееся название ПЗУ — ROM (от англ. read only memory — память только для чтения). Для размещения ПЗУ в адресном пространстве микроЭВМ ему необходимо предоставить часть адресного пространства, свободную от ОЗУ. Пересечение адресов ОЗУ и ПЗУ недопустимо. Наиболее просто отвести для ПЗУ один или несколько банков (рис. 5.23) при наличии схемы селекции банков (рис. 5.24). Для данного примера распределение адресного пространства памяти таково: нижние 16 К байт — ПЗУ, следующие 32 К байт — ОЗУ, верхние 16 К байт — свободная область.

Модульное построение ОЗУ и ПЗУ, возможность быстрого изменения номеров банков позволяют гибко

перестраивать структуру адресного пространства памяти микроЭВМ, перераспределяя его между ОЗУ и ПЗУ в зависимости от требований пользователя и конкретной области применения микроЭВМ.

Применение БИС ЗУ значительно облегчает построение ЗУ по модульному принципу. На рис. 5.25 приведена

0000	ПЗУ
3FFF	
4000	ОЗУ1
BFFF	ОЗУ2
C000	
FFFF	

Рис. 5.23

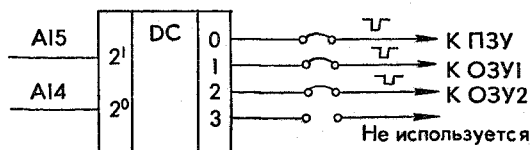


Рис. 5.24

схема модуля ОЗУ емкостью 4 К байт на микросхемах К537РУ8. Одна из перемычек S0 — S15 должна быть замкнута, этим определяется выбор одного из шестнадцати банков. Для замкнутой перемычки, например S1, адреса модуля расположены с 1000_{16} по $1FFF_{16}$. Адресная линия A11 выбирает одну из двух микросхем ЗУ. Для адресов $1000—17FF$ ($A_{11}=0$) выбирается первая БИС ЗУ, а вторая отключена ($CS_1=1$). При адресах $1800—1FFF$ ($A_{11}=1$) наоборот: первая БИС ЗУ отключена ($CS_1=1$), вторая включается при обращении к банку.

Внешняя память. ВЗУ имеет наибольший объем — десятки и сотни мегабайт и самое большое время поиска, записи и считывания информации. В отличие от СОЗУ, ОЗУ (ПЗУ), называемых иногда внутренней памятью, ВЗУ в основном использует магнитные носители информации и не участвует непосредственно в процессе вычислений. Поскольку ВЗУ часто конструктивно выполняется в виде отдельного устройства, его относят к периферийным устройствам ЭВМ. Более подробно ВЗУ рассматривается в гл. 6.

5.6. Интерфейсы микропроцессоров и микроЭВМ

Общие сведения. Передачу информации от внешнего устройства (ВУ) по направлению к процессору называют вводом, а от процессора к внешнему устрой-

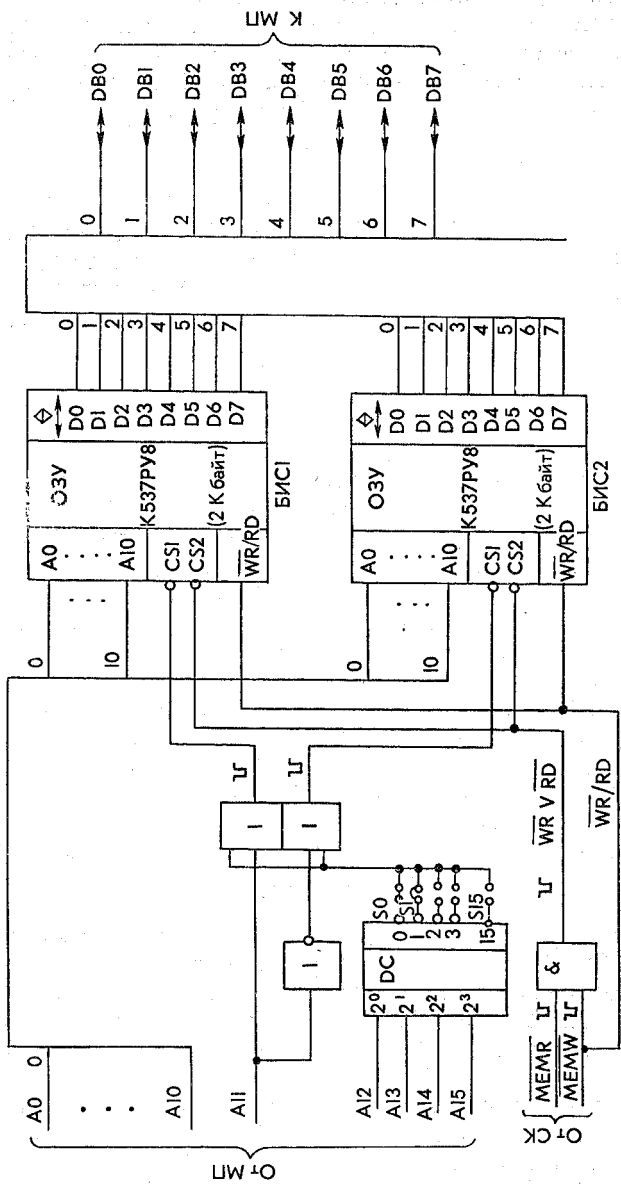


Рис. 5.25

ству — в ыв о д о м. Совокупность унифицированных электрических линий и шин, связывающих устройства микроЭВМ или микропроцессорной системы между собой, а также специальных схем, предназначенных для обеспечения этой связи, алгоритмов (протоколов) обмена дан-

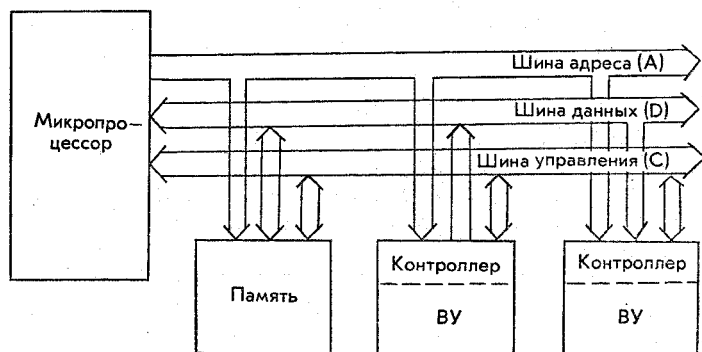


Рис. 5.26

ными, параметров сигналов, участвующих в обмене данными, называют интерфейсом.

В настоящее время наибольшее распространение получили микроЭВМ с магистральным интерфейсом (рис. 5.26). Линии связи между процессором, памятью и внешними устройствами разделяют на три группы (шины): адреса, данных и управления. Электронная схема, входящая в состав ВУ и обеспечивающая подключение к интерфейсу, называется контроллером. Нередко конструктивно контроллер оформлен в виде отдельной платы, которая вставляется в электронный блок микроЭВМ и кабелем соединяется с внешним устройством.

Через контроллер процессор обменивается информацией с ВУ, а также управляет его работой (перематкой ленты, пуском, остановом) и опрашивает его состояние (готовность, наличие ошибок). Обычно в контроллере для этих целей имеются два отдельных регистра: *регистр данных* (РД) и *регистр состояния* (РС).

Каждый разряд регистра состояния имеет свое назначение, определяемое *форматом регистра состояния*. Например, отдельный бит может использоваться для проверки готовности ВУ: если в нем при чтении РС процессор обнаружил 1, значит, ВУ готово к вводу-выводу,

если же в этом разряде 0 — значит, ВУ не готово (например, выполняет предыдущую операцию ввода-вывода).

Так как в системе может быть несколько различных контроллеров, то каждому из них присваивается свой номер (адрес). Если в контроллере имеются РД и РС, то каждому из этих регистров присваивается отдельный адрес. В начале каждой операции ввода-вывода процессор формирует адрес требуемого ВУ, контроллеры анализируют этот адрес, сравнивая его с «собственным» адресом. Контроллер, принявший «свой» адрес, участвует в операции ввода-вывода, остальные контроллеры не мешают ее выполнению.

При организации ввода-вывода в микроЭВМ необходимо решить ряд важных задач, без чего невозможен правильный обмен данными:

1) обеспечить связь только с тем внешним устройством, с которым намерен обмен данными; все другие ВУ не должны мешать «чужому» вводу-выводу;

2) согласовать скорость обмена, так как скорости работы различных ВУ сильно различаются между собой и отличны от скорости работы процессора и памяти. Выделяют группу «медленных» ВУ, быстродействие которых связано с работой механических частей ВУ (печатающее устройство, перфоратор и другие), скоростью работы человека (например, ввод с клавиатуры), и группу «быстрых» ВУ, скорость обмена данными с которыми может приближаться или достигать быстродействия процессора и памяти (например, магнитные диски, аналого-цифровые и цифроаналоговые преобразователи);

3) решить проблему синхронизации обмена, т. е. правильного определения моментов начала и окончания передачи данных между устройствами, участвующими в обмене — передатчиком и приемником данных (без этого нет гарантии правильной передачи информации);

4) согласовать между собой электрические параметры сигналов, которыми обмениваются устройства;

5) обеспечить максимальную скорость обмена минимальными аппаратными и программными средствами.

Противоречивость пятой задачи привела к созданию трех способов обмена данными:

программный ввод-вывод;

ввод-вывод по прерываниям;

ввод-вывод в режиме прямого доступа к памяти.

Программный ввод-вывод. Программный ввод-вывод — это наиболее простой способ обмена данными

между процессором и внешним устройством. Всеми действиями по организации обмена управляет процессор. Перед началом операции процессор должен опросить (проверить) готовность ВУ, а затем, если оно готово, выполнить ввод-вывод. Если же ВУ не готово, процессор должен ждать, периодически опрашивая готовность ВУ (чтение РС). Граф-схема алгоритма программного обмена (фрагмента некоторой программы) приведена на рис. 5.27.

При обмене информацией с медленными ВУ (например, печать со скоростью 10 символов в секунду) процессор вынужден большую часть времени ждать готовности ВУ (в данном случае по 0,1 с на каждый выводимый символ). Например, чтобы напечатать слово ВНИМАНИЕ! (рис. 5.28), процессор должен заниматься этим в течение 1 с, хотя собственно подготовка и передача кода одного символа в РД печати занимает обычно несколько микросекунд, например 10 мкс. Тогда остальные 99 990 мкс, или 99,99 % времени, процессор опрашивает готовность устройства печати. Этот пример показывает, что программный ввод-вывод приводит к простаю процессора, особенно при обмене с медленными ВУ.

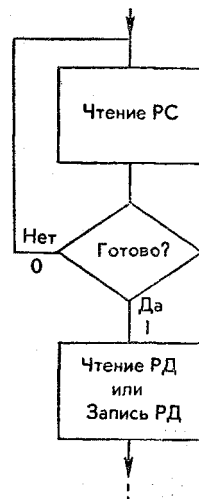


Рис. 5.27

Ввод-вывод по прерываниям. Для повышения производительности системы необходимо освободить процессор от опроса готовности ВУ к обмену. Эта функция возлагается на контроллер ВУ. Получив команду ввода-вывода, контроллер передает ее ВУ и следит за временем ее выполнения. По окончании действий в ВУ (печати, перфорации и т. п.) контроллер посылает в процессор сигнал требования прерывания, получает очередную команду ввода-вывода, и действия повторяются. Процессор в этом случае, передав в контроллер очередную команду ввода-вывода, может выполнять другие операции основной программы до получения сигнала требования прерывания. Получив его, он обслуживает это прерывание, т. е. формирует и выдает в контроллер очередную команду ввода-вывода, а затем возвращается к выполнению прерванной основной программы.

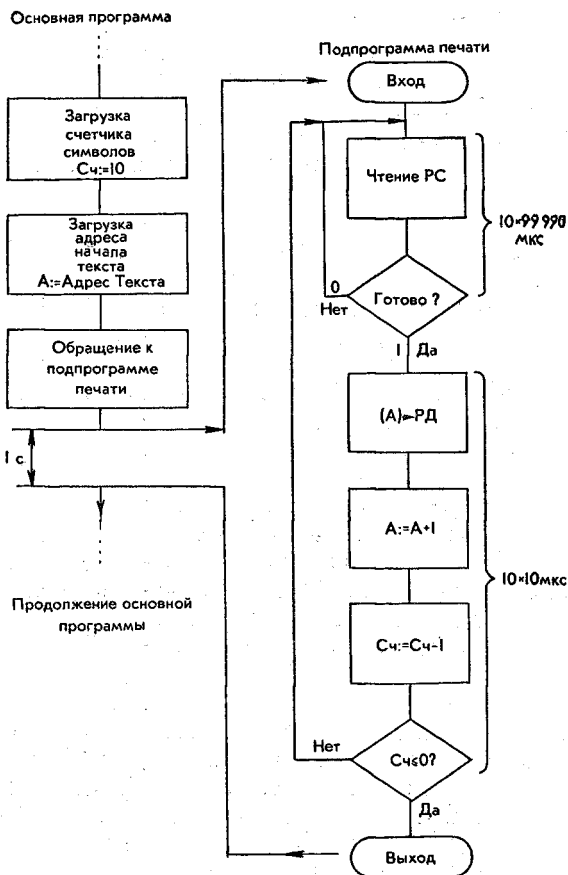


Рис. 5.28

В программе вывода текста, показанной на рис. 5.29, отсутствует ожидание готовности ВУ. Как только устройство печати готово принять очередной символ, контроллер вырабатывает требование прерывания, и на 10 мкс прерывается основная программа, а затем продолжается ее выполнение. Простои процессора сокращаются, так как пока ВУ не готово, процессор выполняет другие команды основной программы, т. е. процессор и ВУ в это время работают параллельно.

Ввод-вывод по прерываниям, однако, требует услож-

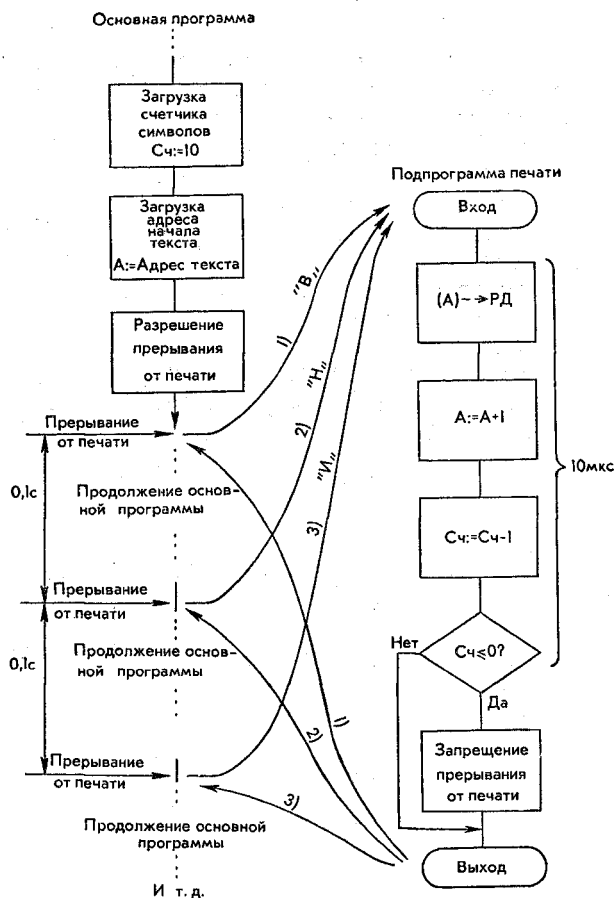


Рис. 5.29

нения аппаратных средств — создания системы прерываний.

Ввод-вывод в режиме прямого доступа к памяти. Два рассмотренных выше способа обмена — программный и по прерываниям — имеют малую скорость обмена данными. Для передачи одного слова данных процессор должен выполнить несколько команд, среди них вспомогательные — изменение адреса памяти, изменение и анализ содержимого счетчика слов. В некоторых микропроцессорах (например, в МП КР580ВМ80) невозможно

в одной команде передать данные из ВУ в память. Сначала необходимо их принять в процессор, а затем из процессора передать в память. Однако во многих случаях требуется передавать большие массивы информации между памятью и внешним устройством (например, накопители на гибких магнитных дисках). В этом случае процессор выступает в роли «лишнего звена», транзитом пропуская через себя информацию. Скорость передачи информации при этом ограничена и не превышает нескольких десятков К байт в секунду.

В то же время память микроЭВМ обычно позволяет выполнять чтение-запись данных со скоростью несколько сот К байт или даже несколько М байт в секунду. Нередко и ВУ позволяют вводить или выводить данные с такими скоростями. Очевидно, что процессор, участвуя в таком обмене, становится «узким местом», снижает возможные скорости обмена, поэтому в режиме прямого доступа к памяти (ПДП) процессор отключается и не участвует в операциях ввода-вывода. Весь обмен информацией выполняется под управлением контроллера ПДП, который «замещает» процессор на это время, т. е. сам формирует адрес памяти, изменяет его и счетчик слов после каждого чтения-записи, принимает решение об окончании обмена и осуществляет синхронизацию.

Контроллер ПДП — сложное устройство. В случае необходимости обмена контроллер сообщает об этом процессору сигналом «Запрос на захват шины» или «Требование прямого доступа». Освободив шины, процессор отвечает сигналом «Подтверждение захвата» («Подтверждение прямого доступа»), и с этого момента контроллер ПДП получает шины интерфейса в свое распоряжение. Режим ПДП позволяет достичь максимально возможной скорости обмена, определяемой физическим быстродействием памяти, ВУ, схем интерфейса. Заметим, что программы обмена в этом случае нет, обмен выполняется аппаратно. Данные передаются из памяти в ВУ и из ВУ в память напрямую, как показано на рис. 5.30.

Прерывания. Как уже отмечалось, для более эффективной организации ввода-вывода в микроЭВМ существует система прерываний. Под прерыванием понимают переход от выполнения основной программы к другой, называемой программой обработки прерывания, по внешнему запросу на прерывание, причем момент появления запроса заранее не известен. Можно провести аналогию между переходом к программе обработки

прерывания и уже известной процедурой обращения к подпрограмме из основной программы. В обоих случаях при выходе из основной программы необходимо запомнить адрес возврата, т. е. адрес команды основной программы, которая будет выполняться после возврата из подпрограммы. Для этого используется стек, куда

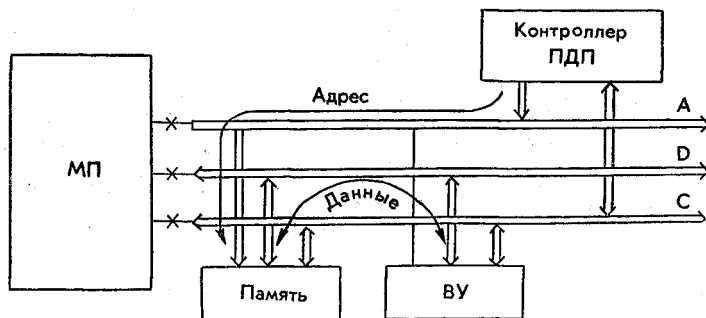


Рис. 5.30

загружается содержимое программного счетчика (адрес возврата) при переходе к подпрограмме или при прерывании. Отличие, как уже отмечалось, состоит в том, что момент обращения к подпрограмме заранее «запланирован» в основной программе записью в нужном месте команды обращения к подпрограмме. В системе команд микропроцессора КР580ВМ80 это команда CALL, в системе команд микроЭВМ «Электроника-60» (ДВК, микропроцессоры К1801ВМ1, ВМ2, ВМ3) — команда /SR.

Если микроЭВМ должна обслуживать несколько устройств, способных требовать прерывание, то при появлении запроса необходимо определить источник прерывания и обеспечить переход на соответствующую программу обслуживания, так как каждое устройство обслуживается по-разному. Таким образом, работу системы прерывания можно свести к трем функциям:

- 1) обнаружение запроса (требования) на прерывание;
- 2) определение источника прерывания;
- 3) переход на соответствующую этому источнику программу обслуживания (с запоминанием адреса возврата).

Указанные функции в каждой конкретной микроЭВМ реализуются по-разному, при этом используется сочетание аппаратных и программных средств.

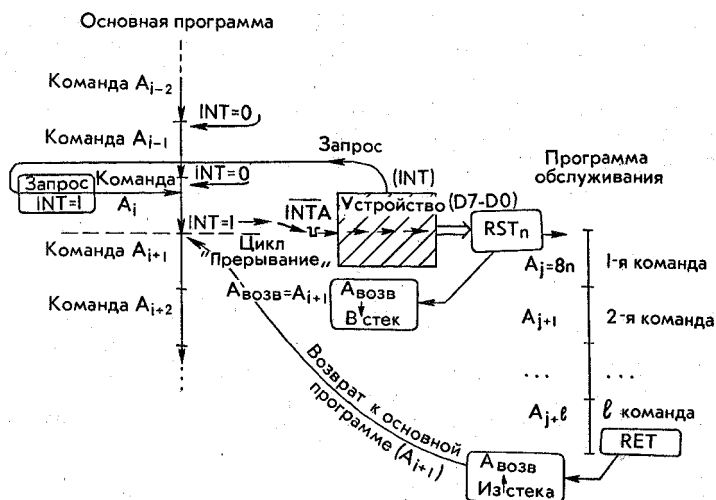


Рис. 5.31

Для выполнения первой функции в наиболее простом случае микропроцессор имеет один вход запроса на прерывание (например, у МП КР580ВМ80 это вход INT).

После выполнения каждой команды микропроцессор проверяет состояние этого входа. Если $INT = 0$ (запроса нет), МП переходит к выборке и выполнению следующей команды. Если же $INT = 1$, т. е. имеется запрос, то МП входит в специальный цикл ПРЕРЫВАНИЕ. При этом системный контроллер (ВК28 или ВК38), работающий совместно с МП, формирует сигнал \overline{INTA} — подтверждение прерывания. Это импульс, в течение которого устройство, потребовавшее прерывания, выдает на шину данных D7 — D0 код команды RST (от англ. restart — рестарт).

Формат команды RST_n однобайтный: $11XXX111$, где $XXX = n$ — номер команды RST.

Всего существует 8 команд RST: RST0, RST1, RST2, ..., RST7. Команда RST_n выполняется следующим образом: в стеке запоминается текущее содержимое программного счетчика PC (адрес возврата), при этом содержимое указателя стека SP уменьшается на 2; в PC загружается новое содержимое, равное $00000000\ 00XXX000$, т. е. выполняется переход на команду с адресом, определяемым командой RST_n . По RST0 — переход на адрес 0000_{16} , по RST1 — на 0008_{16} , по RST2 — на 0010_{16} ,

РС (при этом содержимое указателя стека SP увеличивается на 2). Так осуществляется возврат к прерванной программе и ее выполнение продолжается. Схематично процесс обслуживания прерывания приведен на рис. 5.31, а один из вариантов формирования кода команды RSTn в устройстве, потребовавшем обслуживания,— на рис. 5.32. В данном устройстве переключателями задан код RST5.

Таким образом, система прерываний, используемая в микроЭВМ на базе МП КР580ВМ80, выполняет следующие функции: обнаружение запроса (вход INT), определение источника (INTA→RSTn), переход на программу обслуживания (RSTn→8n).

Для программного запрета и разрешения прерывания в МП КР580ВМ80 имеются команды DI и EI.

5.7. Специальные БИС системы ввода-вывода

КР580ВВ55. Интерфейсные БИС МПК КР580 предназначены обеспечивать эффективный обмен информацией между компонентами МП системы. К ним относятся адаптеры параллельного и последовательного интерфейсов (ВВ55 и ВВ51), программируемый контроллер прерываний ПКП (ВН59) и контроллер прямого доступа к памяти КПДП (ВТ57). Эти БИС поддерживают: программный обмен (ВВ55, ВВ51), обмен по прерываниям (ВВ55, ВВ51, ВН59), обмен между ОЗУ и ВУ в режиме прямого доступа к памяти (ВТ57, шинные формователи).

В качестве примера рассмотрим работу БИС КР580ВВ55 — программируемого адаптера параллельного интерфейса (ПАПИ). Структура ПАПИ приведена на рис. 5.33. Шинной данных, управляющими сигналами, адресными линиями БИС подключается к МП, шинами РА, РВ, РС — к внешним устройствам. Возможны три режима работы БИС:

- 1) режим 0 — основной режим ввода-вывода (не стробированный);
- 2) режим 1 — стробированный ввод-вывод (однонаправленный);
- 3) режим 2 — стробированный двунаправленный обмен.

В режиме 0 каждый порт может работать либо на ввод, либо на вывод. Можно организовать два 8-битных и два 4-битных порта (16 возможных комбинаций

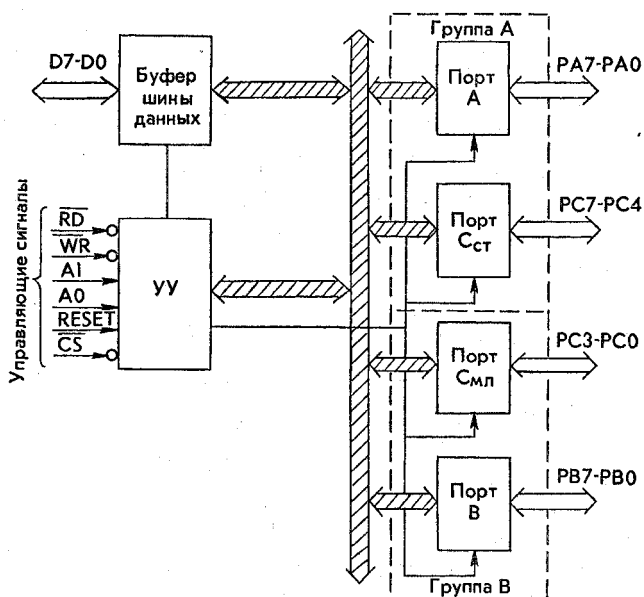


Рис. 5.33

ввода-вывода) либо три 8-битных порта (8 комбинаций ввода-вывода). При выводе (команда OUT) данные из МП запоминаются в регистре адаптера и выдаются на шины порта. При вводе регистры адаптера становятся «прозрачными» для информации из порта (т. е. не фиксируют данные), и по команде IN транслируют данные с шины порта на шину D7 — D0 МП.

В режиме 1 шесть разрядов порта С используются для синхронизации обмена МП с ВУ через порты А и В — по три разряда на каждый порт.

Ввод

STB — сигнал от ВУ, сопровождающий данные. По этому сигналу данные запоминаются во внутреннем регистре адаптера;

OBIBF — «входной буфер заполнен» (данные от ВУ еще не введены в МП) — сигнал от адаптера к ВУ, означающий готовность адаптера к приему новой информации;

INTR — сигнал запроса на прерывание (от адаптера к МП), формируется после приема новой информации из ВУ в адаптер.

Вывод

- \overline{ACK} — сигнал от ВУ, означающий, что ВУ приняло данные;
 \overline{OBF} — «выходной буфер заполнен» (данные из МП переданы в адаптер) — сигнал от адаптера к ВУ, означающий готовность адаптера выдать новую информацию;
INTR — сигнал запроса на прерывание (от адаптера к МП), формируется после принятия информации из адаптера в ВУ. Он позволяет использовать систему прерываний и тем самым повысить производительность микроЭВМ при наличии «медленных» ВУ.

В режиме 2 может использоваться только порт А, а 5 линий порта С являются при этом служебными. Оставшиеся линии порта С и порт В могут использоваться при этом в режиме 0 либо 1.

Кроме указанных режимов, адаптер может выполнять специальные директивы отдельной установки в 0 или 1 любого бита порта С. Директива определяется управляющим словом, загружаемым в адаптер командой OUT.

Если адаптер включается в микроЭВМ, ему присваиваются четыре адреса в адресном пространстве портов ввода-вывода. Например, это могут быть адреса 10, 11, 12, 13 (шестнадцатеричные). При этом адресные линии A1, A0 выбирают один из внутренних регистров адаптера:

- A1 A0
0 0 — порт А — чтение/запись;
0 1 — порт В — чтение/запись;
1 0 — порт С — чтение/запись;
1 1 — управляющее слово — только запись.

Управляющее слово, посылаемое в адаптер, предназначено для настройки адаптера на требуемые режимы работы портов и для отдельного управления разрядами шины С. Форматы управляющих слов приведены на рис. 5.34, а, б.

Для задания группы адресов адаптера используется вход \overline{CS} . Если количество адаптеров не больше шести, то вход \overline{CS} можно подключить к одной из линий A7—A2. При этом адрес адаптера определяется кодом с нулем в этом разряде адреса, например при соединении \overline{CS} с A2 группа адресов адаптера: F8 (А), F9 (В), FA (С) и FB (управление). Если количество адаптеров больше шести, то необходимо ставить дешифраторы, при этом группе адресов будет соответствовать некоторая комбинация

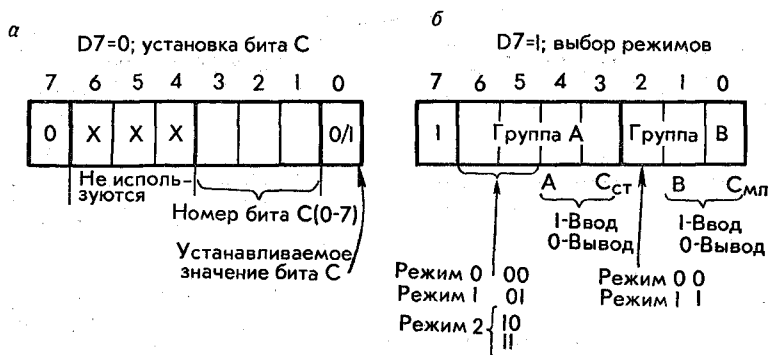


Рис. 5.34

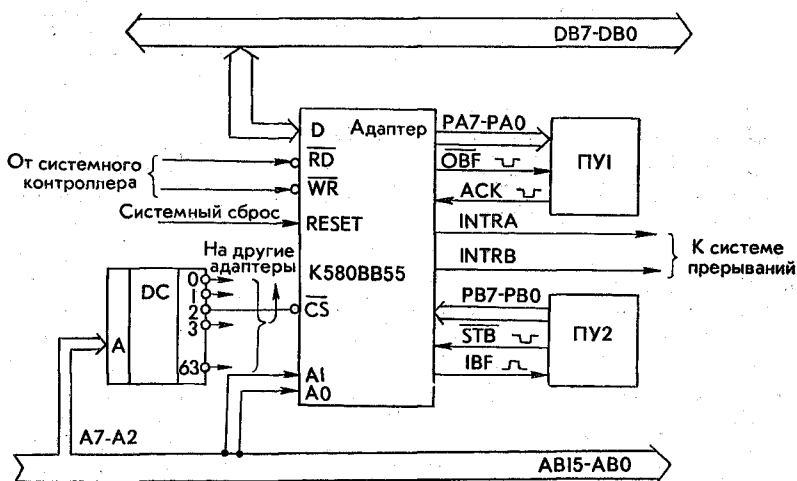


Рис. 5.35

нация A7 — A2. Пример включения адаптера в МП систему приведен на рис. 5.35. Здесь вход CS адаптера подключен к выходу дешифратора, соответствующему комбинации A7 — A2, равной 000010. Следовательно, группа адресов адаптера:

$0000\ 1000_2 = 08_{16}$ (порт A),
 $0000\ 1001_2 = 09_{16}$ (порт B),
 $0000\ 1010_2 = 0A_{16}$ (порт C),
 $0000\ 1011_2 = 0B_{16}$ (управляющее слово).

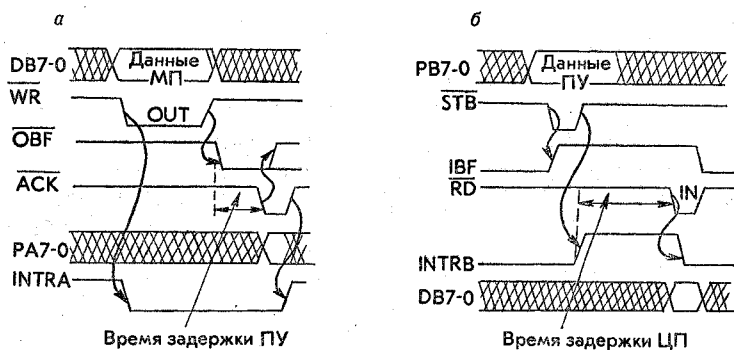


Рис. 5.36

На рис. 5.36, а, б приведены временные диаграммы основных сигналов между адаптером, МП и ПУ. Выводу в порт А соответствует команда OUT 08H, вводу из порта — В — IN 09H. Для того чтобы адаптер работал в приведенной на рис. 5.35 конфигурации (порт А — режим 1, вывод; порт В — режим 1, ввод), необходимо предварительно (до первого обращения к портам) произвести инициализацию адаптера, т. е. передать в него управляющее слово. Код управляющего слова в соответствии с форматом на рис. 5.34 должен быть равен 1010Z11X₂, где Z — управление двумя свободными линиями РС на ввод или на вывод, X — безразличное состояние. Предположим, код 10100110₂ = A6₁₆. Тогда для инициализации адаптера необходимо выполнить две команды:

MVI A, A6H — создание в аккумуляторе кода инициализации;

OUT 0BH — выдача кода в адаптер.

После выполнения этих команд адаптер инициализирован и к нему можно обращаться с командами IN 09H и OUT 08H.

КР580ВН59. БИС программируемого контроллера прерываний КР580ВН59 предназначена для управления 8-уровневыми векторами приоритетными прерываниями. Под управлением МП в БИС могут быть установлены:

- режим обслуживания;
- системы приоритетов;
- векторы прерываний;
- маски запрещения прерываний.

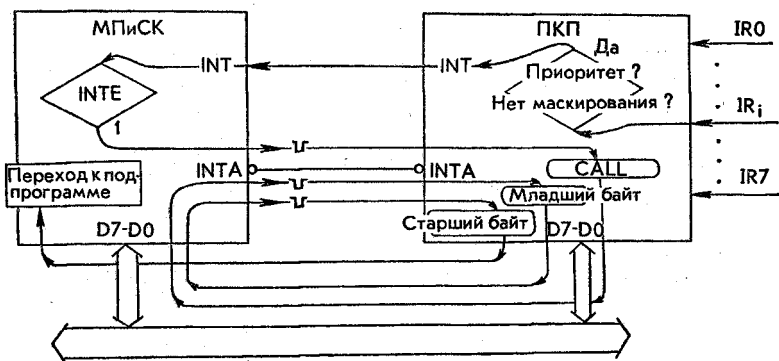


Рис. 5.37

Основная задача, возлагаемая на БИС КР580ВН59,— обнаружить запрос (IR) на прерывание от одного из 8 источников запросов, проанализировать наличие разрешения, приоритета, выдать сигнал запроса (INT) в МП, затем, при появлении из МП сигналов подтверждения (INTA), выработать последовательность кодов (CALL, младший байт и старший байт адреса), обеспечивающую переход к подпрограмме обслуживания именно этого запроса. Схема прохождения этапов обработки запроса от его появления до перехода на соответствующую подпрограмму приведена на рис. 5.37.

КР580ВТ57. БИС программируемого контроллера прямого доступа к памяти (КПДП) предназначена для управления обменом между памятью и ПУ в режиме прямого доступа к памяти.

Режим ПДП предполагает передачу данных из ПУ в память или обратно без участия центрального процессора. При этом используются системные шины данных и адреса. Следовательно, на время обмена ПДП процессор должен освободить эти системные шины, а также линии управления чтением-записью и вводом-выводом. Формирование этих сигналов управления и задание адреса памяти при обмене вместо процессора берет на себя контроллер ПДП (рис. 5.38, 5.39).

БИС КР580ВТ57 позволяет обслуживать до четырех периферийных устройств, управляя передачей блоков до 16 К байт между ОЗУ и ПУ. Канал ПДП устанавливается после запроса от ПУ (DRQ). Если ПДП разрешен, то КПДП посылает в МП запрос HRQ (на вход

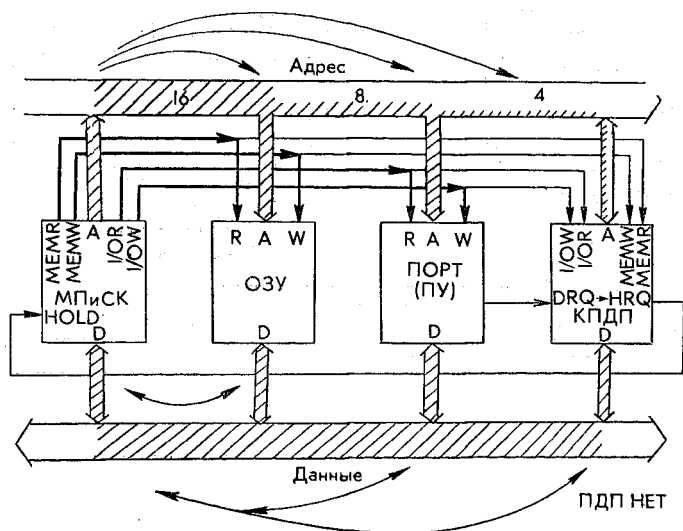


Рис. 5.38

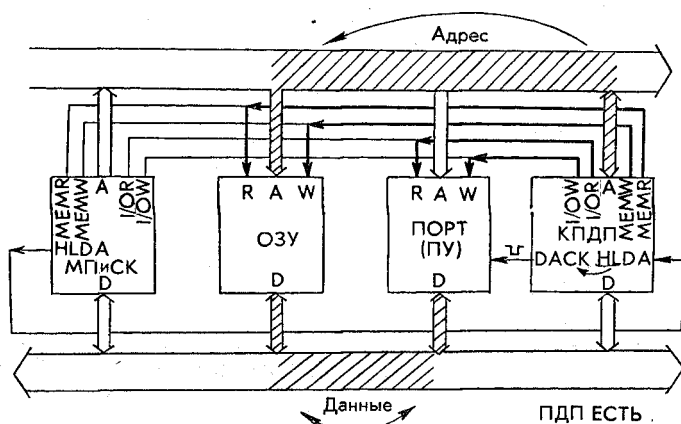


Рис. 5.39

HOLD — «захват»). Когда МП освобождает шины, он переходит в состояние HOLD и выдает сигнал подтверждения HLDA. При появлении данного сигнала в КПДП начинается обмен, ПУ получает при этом подтверждение

DACK. При передаче блоков КПДП автоматически изменяет адрес памяти и следит за окончанием передачи блока. Адрес памяти формирует КПДП, но из-за ограничения на число выводов БИС, старшие разряды A15 — A8 мультиплексируются с данными по шине D7 — D0. Старший байт адреса сопровождается сигналом ADSTB (строб адреса) и должен быть запомнен во внешнем регистре.

Контрольные вопросы и задания

1. Что называется микропроцессором?
2. Расскажите об однокристалльных и многокристалльных микропроцессорах.
3. Нарисуйте функциональное обозначение МП КР580ВМ80. Объясните назначение входов и выходов.
4. Начертите структурную схему микроЭВМ на базе МПК КР580. Объясните работу схемы.
5. Какие функциональные узлы входят в состав ОМП КР580ВМ80?
6. Назовите БИС системы ввода-вывода серии КР580.
7. Какие функциональные узлы входят в состав БИС К1804ВС1?
8. Изобразите структурную схему процессора на базе БИС КР1804. Объясните назначение каждого элемента.
9. Что такое микрокоманда? Интерфейс?
10. Как организуются программный ввод-вывод в микроЭВМ, обмен по прерыванию и прямой доступ?

Глава 6. ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА МИКРОЭВМ

6.1. Классификация периферийных устройств

Устройства, обеспечивающие ввод в микроЭВМ информации и вывод промежуточных и результирующих данных, называются периферийными.

Периферийные устройства различаются между собой принципами построения, выполняемыми функциями, скоростью работы, видом носителя информации и другими параметрами. На рис. 6.1 приведена классификация ПУ по назначению. В соответствии с данной классификацией все ПУ микроЭВМ разделяются на четыре группы: устройства связи с объектом, устройства связи с пользователем, внешние запоминающие устройства, устройства документирования.

Устройства связи с объектом (УСО) предназначены для обмена информацией между микроЭВМ и объектом (станком, двигателем, летательным аппаратом, технологическим и физическим процессом и др.). К этой группе ПУ относятся аналого-цифровые (АЦП) и цифроаналоговые преобразователи (ЦАП); датчики измеряемых величин (скорости, напряжения, углового или линейного перемещения и др.); управляемые усилители, согласующие параметры АЦП (ЦАП) с характеристиками датчиков и объекта; таймеры, задающие временные интервалы для моментов дискретного управления объектом, преобразования и приема информации в микроЭВМ.

Устройства связи с пользователем (УСП) обеспечивают прием информации от человека-оператора и оперативный вывод данных в форме, удобной для восприятия им. С этой целью используются электрические пишущие машинки, пульта управления, клавиатура, дисплей (устройства отображения информации на экране электронно-лучевой трубки, электролюминесцентного индикатора, матричной индикаторной панели), устройства связи с удаленными абонентами (модуляторы-демодуляторы сигналов, телетайпы, абонентские пункты и др.).

Внешние запоминающие устройства (ВЗУ) используются для хранения больших объемов информации. В современных микроЭВМ в качестве ВЗУ применяются накопители на магнитных дисках винчестерского типа, гибких магнитных дисках (НГМД) и кассетных магнитных лентах (НКМЛ). Другие типы ВЗУ (на жестких

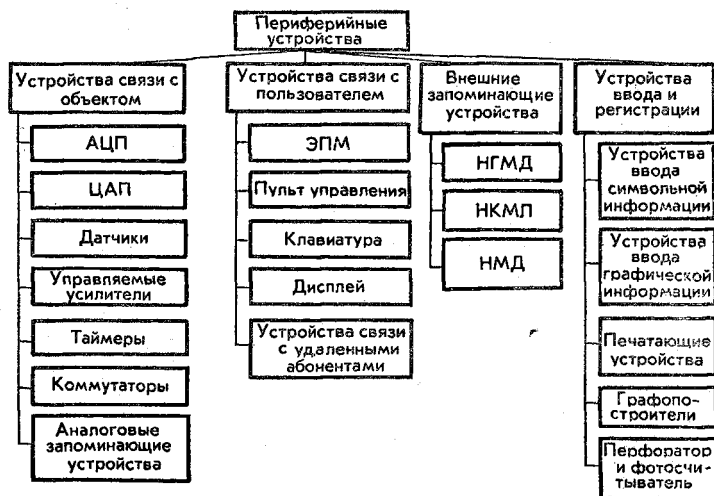


Рис. 6.1

магнитных дисках, барабанах и др.) характерны для больших и мини-ЭВМ, поэтому здесь не рассматриваются.

К *устройствам ввода и регистрации* относятся устройства автоматического считывания символьной и графической информации с какого-либо носителя (перфолен-ты, перфокарты, бумаги) и вывода данных в форме, удобной для восприятия человеком.

В настоящее время разрабатываются новые виды ПУ, обладающие высоким быстродействием, большой плотностью записи информации, возможностью осуществлять предварительную обработку информации с целью разгрузки микроЭВМ. К таким ПУ относятся, например, ВЗУ на цилиндрических магнитных доменах, голографические запоминающие устройства, периферийные процессоры и др.

Независимо от принадлежности к типу ЭВМ периферийным устройствам присвоена единая система индек-

саци. Каждое ПУ обозначается четырехзначным числом, старшая цифра которого указывает принадлежность устройства к определенной группе: внешним запоминающим устройствам — 5; устройствам ввода — 6; устройствам вывода — 7; абонентским пунктам — 8. Три младшие цифры определяют порядковый номер ПУ в группе, а буквенное обозначение, стоящее перед числом, указывает, для какого класса машин разрабатывалось устройство. Например, ЕС-5088 — накопитель на гибком магнитном диске, разработанный для семейства ЭВМ Единой Системы; СМ6304 — алфавитно-цифровое печатающее устройство, разработанное для семейства малых ЭВМ.

Состав периферийного оборудования в конкретной микроЭВМ определяют исходя из удобства работы с ЭВМ, требуемых функций, показателей надежности, быстродействия, объема ВЗУ и др.

6.2. Устройства связи с объектом

Одноканальные УСО. Устройства связи с объектом применяются в микропроцессорных системах, используемых для управления объектами или контроля их параметров. В составе УСО можно выделить две подсистемы: подсистему аналогового ввода и подсистему аналогового вывода (рис. 6.2).

Подсистема аналогового ввода одноканального УСО осуществляет преобразование аналоговой физической величины, характеризующей состояние объекта, в цифровой код и передает его в МПС. При помощи датчика физическая величина преобразуется в пропорциональный ей ток или напряжение, подаваемые на вход *согласующего усилителя* (СУ). Основной функцией СУ является согласование диапазона выходного напряжения (тока) датчика с диапазоном входного напряжения последующих элементов подсистемы. Если датчик имеет высокое выходное сопротивление, то СУ служит также для согласования его с элементами, имеющими низкое входное сопротивление.

Блок фильтрации (БФ) предназначен для фильтрации сигналов от высокочастотных, низкочастотных помех или выделения из спектра сигнала необходимой полосы частот.

С выхода БФ аналоговый сигнал поступает на вход *аналогового запоминающего устройства* (АЗУ). Напряжение на его выходе пропорционально сигналу на входе

до тех пор, пока не последует команда запоминания, после чего выходной сигнал остается постоянным до окончания действия команды. Длительность выполнения команды определяется временем преобразования АЦП, по истечении которого в блок сопряжения (БС) поступит цифровой код, характеризующий измеряемую величину.

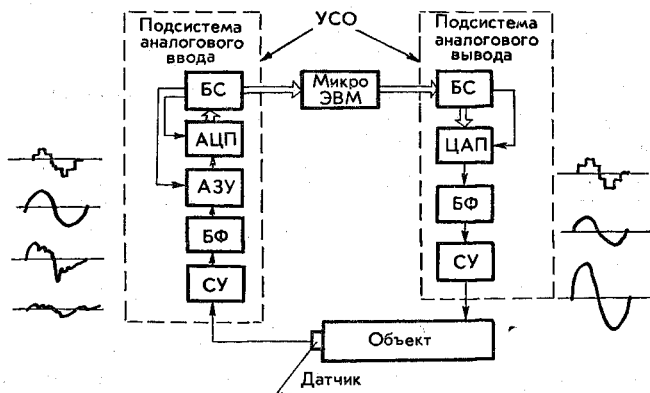


Рис. 6.2

Время преобразования АЦП составляет от десятков наносекунд до сотен миллисекунд, а диапазон входного напряжения, как правило, ± 5 или ± 10 В.

БС формирует управляющие сигналы и организует передачу цифрового кода (либо группы кодов после предварительного накопления информации) в микроЭВМ в программном режиме, режиме прерывания или прямого доступа.

Подсистема аналогового вывода служит для преобразования последовательности цифровых кодов в аналоговый сигнал управления исполнительными устройствами объекта.

Последовательность кодов, принимаемых БС из МПС, поступает в ЦАП, на выходе которого формируется ступенчатое приближение требуемого изменения выходной величины. Большинство ЦАП, как и АЦП, работает в диапазоне напряжений ± 5 или ± 10 В.

После низкочастотной фильтрации, осуществляемой БФ, сигнал усиливается или ослабляется в зависимости от требований, предъявляемых ко входу управления объектом.

Трудности при построении УСД возникают в основном из-за необходимости обеспечения высокой точности, скорости и достоверности при передаче и преобразовании сигналов в условиях наличия помех. Особенно сложно выполнить указанные требования для низковольтных (до 100 мВ) электрических сигналов, характерных для

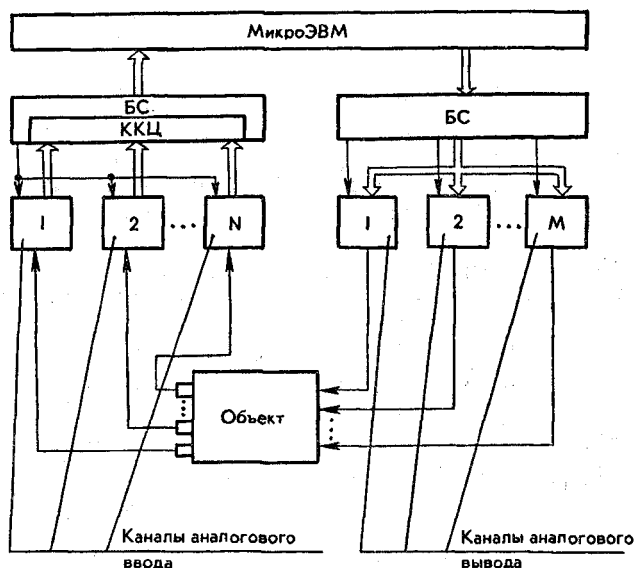


Рис. 6.3

большинства датчиков. Дополнительные сложности возникают при сопряжении МПС с большим числом датчиков (от десятков до тысяч) и десятками исполнительных органов.

Многоканальные УСО. В структуре многоканального УСО появляются дополнительные элементы — коммутаторы аналоговых (ККА) и цифровых (ККЦ) каналов.

Можно выделить три широко распространенные конфигурации подсистем аналогового ввода (вывода): 1) с ЦАП (АЦП) в каждом выходном (входном) канале; 2) с одним ЦАП (АЦП), работающим в режиме разделения времени; 3) комбинированную.

Первая конфигурация представляет собой совокупность модулей аналогового ввода (вывода), имеющих общие блоки сопряжения с микроЭВМ (рис. 6.3). По

командам микроЭВМ блок сопряжения подсистемы аналогового вывода принимает данные и последовательно записывает их в регистры ЦАП первого, второго и последующих до M -го канала. Канал представляет собой цепь ЦАП — БФ — СУ. Соответственно на выходе каждой подсистемы удерживается сигнал до тех пор, пока

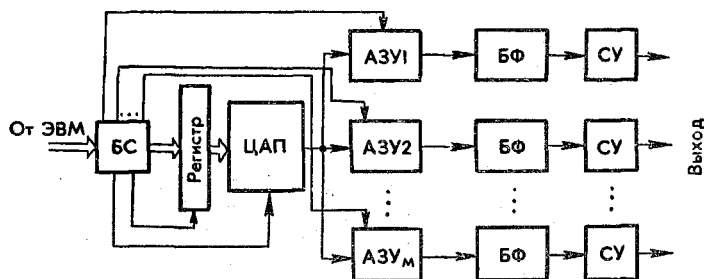


Рис. 6.4

из БС не придет новая команда на запись. Аналогичным образом сигналы с первого, второго и последующего N -го датчика поступают на вход соответствующего канала ввода аналоговой информации (цепь СЦ — БФ — АЗУ — АЦП), преобразуются в цифровые отсчеты, удерживаемые на входах ККЦ до тех пор, пока блок сопряжения не организует их последовательный опрос и передачу в микроЭВМ. После этого из БС поступает команда запуска АЦП для получения следующей группы цифровых отсчетов.

Такая организация УСО используется там, где требуются высокие точность и частота дискретизации входных (выходных) сигналов. Однако, если число каналов велико, такая конфигурация УСО оказывается слишком дорогостоящей из-за сравнительно высокой стоимости АЗУ, АЦП и ЦАП.

Этот недостаток устраняется в структуре многоканального УСО *второй конфигурации* — с одним ЦАП (АЦП). Рассмотрим подсистему аналогового вывода (рис. 6.4). Цифровые коды выходных сигналов 1, 2, ..., M -го каналов последовательно записываются в регистр и подаются на ЦАП, формирующий аналоговый эквивалент соответствующего кода. После окончания преобразования аналоговое значение записывается в соответствующие АЗУ и через БФ и СУ поступает на выход.

Поскольку напряжение на выходе АЗУ с течением времени уменьшается, темп возобновления данных в АЗУ должен быть таким, чтобы снижение напряжения было минимальным.

В структуре подсистемы аналогового ввода с одним АЦП (рис. 6.5) аналоговые сигналы поступают на входы

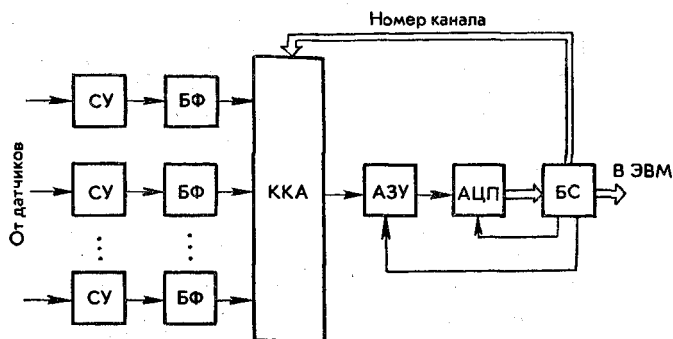


Рис. 6.5

ККА, последовательно коммутирующего на вход АЗУ, а затем и АЦП, входные каналы ККА по управляющему коду из БС. Структуры УСО с одним ЦАП (АЦП) наиболее экономичны по затратам оборудования, однако, поскольку пропускная способность цепочки ККА, АЗУ, АЦП или ЦАП, АЗУ ограничена временем установления (преобразования) соответствующих элементов, эти структуры могут применяться только для управления и анализа сравнительно медленно меняющихся процессов. Вследствие этого довольно часто используют *третью конфигурацию* подсистем — комбинированную, которая представляет собой различные сочетания двух вышерассмотренных и удовлетворяет всем необходимым требованиям.

В состав комбинированных УСО могут входить средства передачи и приема дискретных сигналов, воздействующих на объект или принимаемых из него непосредственно, без предварительного цифроаналогового или аналого-цифрового преобразования. К этим средствам в основном относятся цифровые СУ и ККЦ. При этом схема цифрового СУ оказывается проще схемы аналогового, поскольку передаваемая величина принимает только два значения (0 и 1), соответствующие определенным уровням напряжения.

При сопряжении с распределенными объектами (технологическими линиями, группами установок и др.) возникают дополнительные трудности, вследствие того что система связи с датчиками оказывается многопроводной с большим числом линий связи. Поэтому в состав УСО включают выносные коммутаторы для отдельных групп датчиков, от которых сигналы передаются на центральный коммутатор, т. е. выполняется двухступенчатая коммутация сигналов. Это приводит к некоторому увеличению объема используемого оборудования в системе и снижению скорости передачи сигналов, но существенно упрощает кабельные связи.

В качестве примера рассмотрим структуру УСО, организующего связь процессора ДВК-3 с 16 камерами тепла, холода и влаги. Пусть необходимо обеспечить дистанционное управление включением одного из трех нагревателей, вентилятора, компрессора для подачи хладагента и увлажнителя воздуха в каждой камере в зависимости от значений двух терморезисторов. Для проверки камеры и исследования помещенного в нее изделия требуются, кроме того, показания до 32 терморезисторов или термодпар.

По выходным каналам к каждой камере передаются шесть унитарных цифровых сигналов включения (выключения) соответствующих реле нагревателей, вентилятора, компрессора и увлажнителя. Входные каналы — низковольтные (до 10 мВ) аналоговые. Скорость изменения аналоговой величины не более 1 град/с. Таким образом, в микроЭВМ необходимо передать информацию по 512 аналоговым, а формировать выходные воздействия по 96 цифровым каналам. Большая инерционность исполнительных устройств объекта позволяет использовать схему УСО с одним АЦП (см. рис. 6.5). Для снижения количества линий связи используется двухступенчатая коммутация входных сигналов и последовательная передача цифровых управляющих сигналов. Импульсом из БС счетчики сбрасываются. После этого блоком сопряжения организуется последовательное инкрементирование (увеличение содержимого) счетчиков, опрос входных аналоговых сигналов первой камеры, затем второй, третьей и т. д., преобразование их в цифровой код и передача в микроЭВМ. Одновременно параллельный код управления исполнительными элементами соответствующей камеры при помощи регистра сдвига РСдв преобразуется в последовательный код и передается в РСдв, установ-

ленный вблизи камеры, где выполняется обратное преобразование и запись в регистр Р кода управления исполнительными элементами. В схеме (рис. 6.6) с целью упрощения не показаны передающие и приемные усилители, устанавливаемые на обоих концах линии связи для повышения помехоустойчивости. Рассмотренная

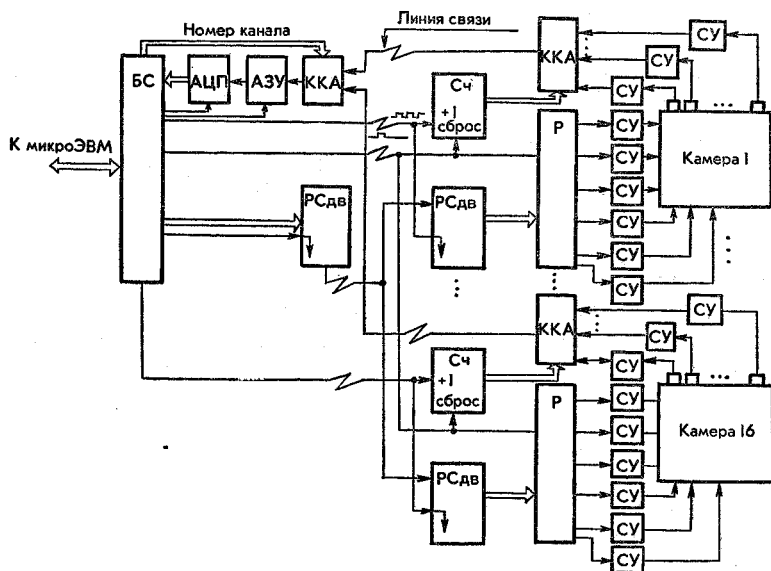


Рис. 6.6

структура УСО имеет лишь четыре линии связи с каждой камерой вместо 38, т. е. позволяет почти в 10 раз сократить число соединительных проводов. Кроме того, в структуре устройства отсутствуют блоки фильтрации. Они оказываются избыточными, поскольку информация с датчиков — статическая.

Для иллюстрации других возможных способов организации УСО рассмотрим модули УСО микроЭВМ «Электроника-60».

Модуль аналогового вывода 15КА-60/4-009. Модуль выполнен на печатной плате. Он содержит блок сопряжения, четыре независимых десятиразрядных ЦАП с регистрами, изолирующий источник питания, источник опорного напряжения и схему гальванической развязки цифровой части модуля от аналоговой и цифроаналоговой. Такая развязка применяется для снижения уров-

ны помех, создаваемых в аналоговых цепях при переключении цифровых элементов.

Ввод цифровых данных в модуль осуществляется по интерфейсу «Общая шина» микроЭВМ «Электроника-60» в программном режиме путем записи информации по 4 адресам, присвоенным модулю.

Техническая характеристика модуля 15КА-60/4-009

Габаритные размеры, мм	252×296×12
Масса, кг	0,5
Диапазон выходного напряжения, В	10
Выходной ток, мА	Не более 5
Разрешающая способность, мВ	20
Абсолютная погрешность преобразования	0,5 % от полной шкалы
Выходное сопротивление, Ом	Не более 1
Время установления выходного напряжения, мкс	Не более 10
Число выходных линий для вывода аналоговой информации	4
Время непрерывной работы, ч	Не менее 16

Модуль аналогового ввода 15КА-60/8-010. В основу работы модуля положен принцип аналого-цифрового преобразования методом поразрядного уравнивания. Модуль выполнен на печатной плате и содержит блок сопряжения, схему гальванической развязки, регистры входных, выходных данных и состояния, изолирующий источник питания и источник опорного напряжения, коммутатор аналоговых каналов, усилитель сигнала с выхода коммутатора, АЦП, схему определения полярности.

Модуль имеет выход на интерфейс «Общая шина» микроЭВМ «Электроника-60». Запуск АЦП осуществляется по программе микроЭВМ путем записи в регистр вывода 4-разрядного кода входного аналогового канала. Окончание преобразования индицируется единичным значением 10-го разряда регистра состояния. Цифровой код принимается из 11-разрядного регистра ввода (10 разрядов данных и один знаковый).

Техническая характеристика модуля 15 КА-60/8-010

Габаритные размеры, мм	252×296×12
Масса, кг	0,5
Диапазон входных напряжений, В	±10 или ±1
Входное сопротивление каждого канала, МОм	Не менее 1

Входная емкость каждого канала, пФ	Не более 100
Входной ток каждого канала, нА	Не более 0,12 % от полной шкалы
Время преобразования, мкс	Не более 100
Число однопроводных линий аналогового ввода	16
Число дифференциальных линий аналогового ввода	8
Время непрерывной работы, ч	Не менее 16

6.3. Устройства связи с пользователем

Общие сведения. Устройства связи с пользователем предназначены для оперативного ввода-вывода информации из микроЭВМ, задания различных режимов ее работы, опроса отдельных ячеек памяти, задания места загрузки программ и данных. Эти устройства обеспечивают непосредственную связь человека с ЭВМ и поэтому им обычно присваивается высший приоритет. Это означает, что при нескольких одновременно поступивших запросах на обслуживание микроЭВМ выбирает в первую очередь запрос от УСП. Время реакции на такой запрос обычно не превышает 1,5—2 с, у пользователя создается иллюзия монопольного режима обслуживания и практически мгновенной реакции системы на запрос, поэтому УСП часто называют интерактивными средствами взаимодействия.

Наверное, ни для одного класса ПУ не существует столь много разновидностей аппаратных и программных средств, как для УСП. Устройства связи с пользователем определяют язык и форму общения человека с системой, степень полноты и достоверности получаемой и вводимой информации, скорость обмена данными, время расшифровки (осмысления) сообщения и т. п.

В первых микропроцессорных системах для организации интерактивного диалога в основном применялись электрические пишущие машинки (ЭПМ) ударного действия (рис. 6.7) с набором готовых знаков. Печатающий механизм такой ЭПМ состоит из рычага, закрепленного на оси и имеющего на конце литеру (пластинку с рельефным изображением знака). По коду знака, подаваемому из микроЭВМ, на одном из выходов дешифратора формируется импульс тока, под действием которого сердечник втягивается в соленоид, рычаг поворачи-

чивается вокруг оси и через красящую ленту ударяет по бумаге. После прекращения действия импульса рычаг возвращается пружиной в исходное состояние. Перемещение бумаги происходит за счет барабана.

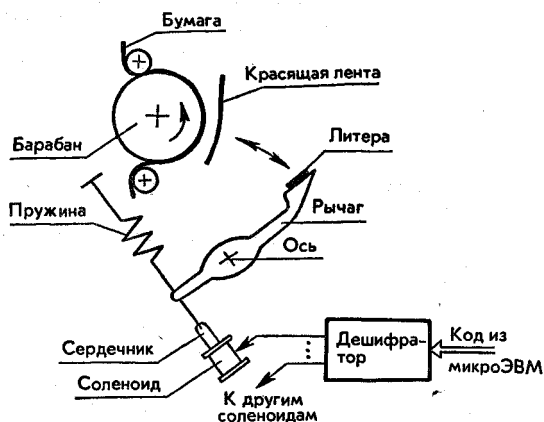


Рис. 6.7

Быстродействие таких устройств крайне низкое — 40—50 ударов в секунду, что существенно ограничивает объем вводимых и выводимых сообщений. Кроме того, сравнительно быстрое изнашивание механических частей резко снижает срок службы и надежность ЭПМ.

Поиск устройств, устраняющих недостатки ЭПМ, привел к созданию современных устройств ввода и отображения информации на экране электронно-лучевой трубки (ЭЛТ) — дисплеев.

Дисплей можно классифицировать по различным признакам (рис. 6.8). По способу отклонения луча различают дисплеи с *функциональной разверткой (векторные)*, в которых электронный пучок, засвечивающий слой люминофора электронно-лучевой трубки, отклоняется по заданной программе (например, подобно лучу осциллографа); с *растровой разверткой (телевизионные)*, в которых электронный пучок последовательно сканирует поле экрана, организуя в определенные моменты его засветку или затемнение.

По назначению дисплеи подразделяются на *универсальные*, способные функционировать в составе аппаратуры широкого применения, и *специализированные*,

ориентированные на выполнение узкого круга задач.

По способу кодирования информации дисплеи можно разделить на *алфавитно-цифровые*, в которых диалог с системой ведется посредством текста; *графические*, в которых информация представляется в виде графиков, таблиц, схем; *рельефные*, формирующие сти-



Рис. 6.8

лизованные объемные образы различных объектов (шар, куб и т. д.). В алфавитно-цифровых дисплеях можно также формировать простейшее графическое изображение путем соответствующей конфигурации псевдографических символов. В графических дисплеях, в свою очередь, есть возможность отображать алфавитно-цифровую информацию.

В соответствии со способом представления данных различают дисплеи *монохроматические*, *цветные*, *полутоновые* (с большим числом оттенков).

Благодаря сравнительно низкой стоимости, высокой надежности и плотности изображения, а также возможности использования телевизионного индикатора растровые монохроматические дисплеи получили наиболее широкое распространение в МПС. В таком дисплее поле экрана ЭЛТ (монитора) условно разбивается на $V \times H$ -элементов изображения (V — число точек телевизионной (ТВ) строки, H — число ТВ-строк), сканируемых

по траектории, показанной на рис. 6.9. Совокупность светящихся точек раstra образует видимое изображение. Чтобы изображение было немерцающим, его необходимо регенерировать (возобновлять) с частотой $F_p = 45 - 50$ Гц (период $T = 20 - 25$ мс). Для формирования знаковой информации поле экрана дополнительно раз-

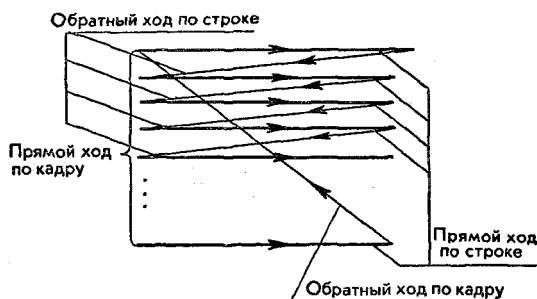


Рис. 6.9

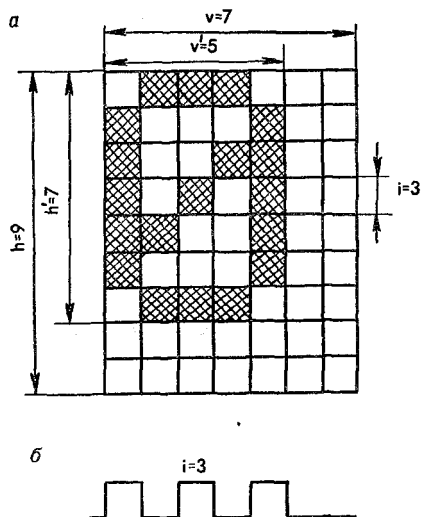


Рис. 6.10

бывают на знакоместа, содержащие $v' \times h'$ элементов изображения или $v \times h$ — с учетом затемняемых точек и строк между символами (рис. 6.10).

Отображение знаковой информации. В период прямого хода луча по строке и по кадру из оперативного запоминающего устройства регенерации (ОЗУР) последовательно считываются коды отображаемых символов (рис. 6.11, а), расположенные таким образом, что каждой ячейке ОЗУР соответствует позиция на экране ЭЛТ.

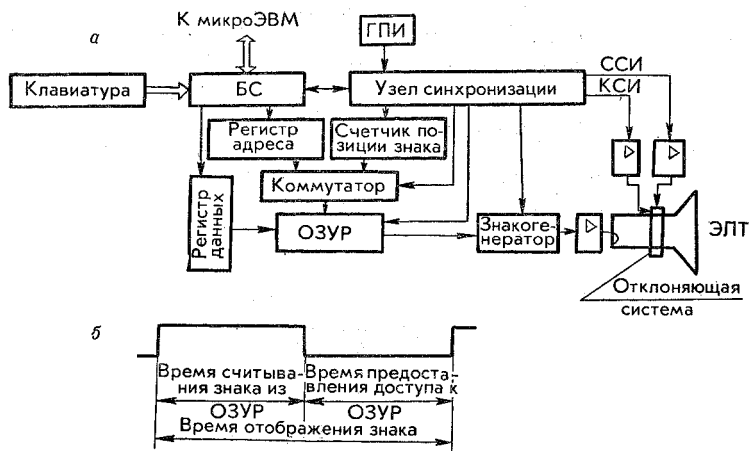


Рис. 6.11

На время считывания кода знака к адресным входам ОЗУР подключается выход счетчика позиции знака, содержимое которого увеличивается на единицу всякий раз при переходе к следующей позиции знака. Считанный из ОЗУР код знака поступает в знакогенератор, где по коду текущей строки разложения знака $i = 0, 1, \dots, h - 1$ (рис. 6.11, б) формируется последовательный код модуляции луча ЭЛТ по яркости. Например, для $i = 3$ сформируется сигнал, форма которого показана на рис. 6.10, б. В период обратного хода по строке счетчик позиции знака возвращается в исходное состояние, соответствующее нулевой позиции в текущей знаковой строке. Если очередная телевизионная строка участвует в формировании последующей знаковой строки, то содержимое счетчика позиции знака в период обратного хода по строке установится в состояние, соответствующее нулевой позиции знака в очередной знаковой строке.

В период обратного хода по кадру счетчик позиции

знака обнуляется, что соответствует левому верхнему углу поверхности экрана. Описанные процессы протекают синхронно с движением луча по экрану монитора. Управление отклонением луча достигается строчным (ССИ) и кадровым (КСИ) синхронизирующими импульсами, запускающими соответствующие генераторы развертки по строке и кадру. Временное расположение всех управляющих импульсов задается узлом синхронизации, работающим с тактовой частотой 15—20 МГц, формируемой кварцевым генератором прямоугольных импульсов (ГПИ). Требования к стабильности ГПИ очень высоки, поскольку малейшее изменение частоты может вызвать «размытие» или «подрагивание» изображения.

Минимальная тактовая частота ГПИ при заданных V (с учетом невидимой части изображения в период обратного хода по строке) и периоде ССИ T_c ($T_c = 64$ мкс) равна $F_{\min} = V/T_c$, а период считывания знаков из ОЗУР составляет $T_{\text{ОЗУР}} = v/F_{\min}$. Полное число знаков в строке $N_c = T_c/T_{\text{ОЗУР}}$, в видимой части соответственно $N_{\text{св}} = N_c(1 - \alpha)$, где α — коэффициент, показывающий, какую часть от T_c составляет видимая часть ТВ-строки. Обычно $\alpha = 0,2—0,3$.

Полное число ТВ-строк H ограничено величиной $H < 1/(F_p T_c)$ с целью получения немерцающего изображения, а число знаковых строк $N_{\text{з.с}} = H/h$. В видимой части кадра $N_{\text{з.св}} = N_{\text{з.с}}(1 - \beta)$, где β — коэффициент обратного хода по кадру, равный 0,05—0,07.

Минимальный объем ОЗУР в битах равен соответственно: $Q = N_{\text{св}} N_{\text{з.св}} N_{\text{я}}$, где $N_{\text{я}}$ — разрядность ячейки, зависящая от алфавита знаков и возможных признаков отображения символа на экране, указывающего на необходимость его выделения по яркости мигания, подчеркивания, негативного изображения и т. п.

Например, при $F_p = 50$ Гц, $V = 640$, $H = 312$, $v = h = 8$ минимальная тактовая частота составит $F_{\min} = 10$ МГц; $T_{\text{ОЗУР}} = 0,8$ мкс; $N_c = 80$, $N_{\text{з.с}} = 39$.

Запись информации в ОЗУР дисплея может быть организована тремя способами: в режиме независимого, последовательного и прямого доступов.

В режиме независимого доступа по инициативе микроЭВМ происходит останов регенерации изображения, к адресным входам ОЗУР подключается регистр адреса и организуется заполнение ОЗУР новыми данными. Такой метод обладает малым временем обмена информацией $T_{\text{об}}$, но приводит к нежелательным эффек-

там пропадания или мерцания изображения, особенно при большом числе данных и высокой частоте обращения к дисплею.

В режиме последовательного доступа запись данных в ОЗУР осуществляется в период обратных ходов или во время затемнения между знаковыми строками. Этот способ прост, не создает мерцаний и наиболее широко используется в современных дисплеях, но характеризуется большим временем ожидания готовности ОЗУР к приему данных. Общее время, предоставляемое ОЗУР для связи с микроЭВМ, $T_{об}$ составляет около 19 %. Современные интегральные запоминающие устройства позволяют увеличить $T_{об}$ до 50 % за счет использования для обмена данными времени, остающегося после считывания кода символа в знакогенератор (рис. 6.11, б).

В режиме прямого доступа дисплей осуществляет прием информации из микроЭВМ без прерывания регенерации изображения в период обратных ходов или затемнений. Этот режим наиболее благоприятен для дисплея, но ограничивает возможности микроЭВМ и усложняет блок сопряжения.

Отображение графической информации. На телевизионном растре можно воспроизвести графическую информацию с разрешающей способностью не более чем геометрические размеры растровой точки.

Графическая информация может представляться в виде набора псевдографических символов (крестиков, звездочек, штрихов и т. п.). В этом случае графическое изображение разбивается на одинаковые элементарные участки (знакоместа без промежуточных затемнений), а каждый такой участок соответствует определенному символу в алфавите псевдографических символов. Алфавит псевдографических символов состоит из набора отрезков прямых под различными наклонами к линии растра, сплошных частичных «заливок» поля графического символа и т. д. Принципы формирования графического изображения при таком подходе аналогичны формированию знаковой информации. Но при этом усложняется программирование графического рисунка, поэтому существует ряд специальных способов формирования графиков. Например, если повернуть растр дисплея на 90° , то формирование однозначного графика сведется к осуществлению засветки отрезка телевизионной строки длиной, пропорциональной коду,

считанному из ЗУ. На этом основан принцип работы схемы, изображенной на рис. 6.12.

Генератор графика состоит из регистра, счетчика точек, схемы сравнения и элемента И. В течение обратного хода предыдущей строки код ординаты графика принимается в регистр. Одновременно обнуляется счет-

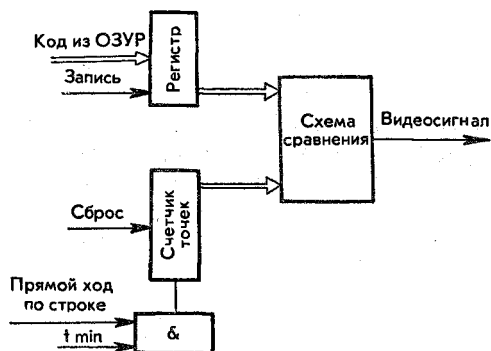


Рис. 6.12

чик точек. С началом прямого хода очередной строки запускается счетчик точек, и когда его код совпадает с кодом, хранящимся в регистре, схема сравнения выдает сигнал окончания засветки строки или засветки в этот момент точки. В первом случае график получается в виде вертикальных светящихся линий, во втором — в виде точек.

Аналогичным способом формируется график при нормальной, горизонтальной развертке, с той лишь разницей, что считывание из ЗУ символов происходит в темпе перехода луча от одной точки по растру до другой. Требуемое время считывания составляет 62—120 нс и накладывает ограничения на ЗУ.

В персональных ЭВМ (ПЭВМ) широко распространен метод формирования графиков, основанный на кодировании полнографической информации. Согласно этому методу, объем ОЗУР составляет $Q = VHN$, (N — двоичный логарифм числа градаций по яркости), графическое изображение по точкам программируется в микроЭВМ, а затем выводится в дисплей. Метод достаточно универсален для большого класса изображений, но обладает по сравнению с рассмотренными рядом недостатков: требует высокого времени формирования

и вывода изображения, большого объема памяти микроЭВМ.

Пульт управления. Используется для индикации состояния системы и отдельных ее функциональных частей, организации пошагового или потактного выполнения команд, проведения контрольных и диагностических работ. Однако по мере повышения степени интеграции элементной базы непосредственная индикация многих элементов микроЭВМ (например, регистров микропроцессора) становится затруднительной. Функции, ранее выполняемые пультом управления, переносятся на микропрограммы и программы. Вместо традиционных кнопок и переключателей используется клавиатура, а в качестве элементов индикации — дисплей или ЭПМ.

Рассмотрим пульт управления микроЭВМ «Электроника-60», имеющий три переключателя режима работы, универсальную алфавитно-цифровую клавиатуру и соответствующие светодиодные индикаторы: «Вкл/выкл питания»; «Программный/пошаговый режим»; «Вкл/выкл таймера».

Остальные функции (команды) пульта управления (табл. 6.1) организуются специальными микропрограммами, хранящимися в ПЗУ микроЭВМ. Чтобы прочесть содержимое ячейки памяти с адресом 200₍₈₎, оператор в пошаговом режиме должен набрать на клавиатуре адрес и затем нажать на клавишу «/». Код этой клавиши

Таблица 6.1

Номер п/п	Символ	Код	Содержание команды
1	2	3	4
1	/	017	Открыть ячейку. Отображение содержимого ячейки памяти
2	BK	015	Закрыть ячейку. Окончание работы с выбранной ячейкой памяти
3	PC	012	Закрыть ячейку и открыть следующую
4	[136	Открыть предыдущую ячейку
5	@	100	Открыть ячейку с абсолютным адресом
6	@_	137	Открыть ячейку с относительным адресом
7	R(α)	122 (044)	Регистр, используется для обращения к регистрам общего назначения
8	RS	122 и 123	Слово состояния процессора

1	2	3	4
9	G	107	Пуск. Пуск программы со стартового адреса
10	P	120	Продолжение. Продолжение выполнения программы
11	M	115	Отладка
12	ЗБ	177	Забой. Исправление последнего введенного символа
13	L	114	Загрузка. Ввод программы абсолютно-го загрузчика

является начальным адресом микропрограммы, которая считывает из памяти ячейку с адресом 200 и выводит ее содержимое на экран дисплея после знака «/»:

@ 200/177777

Знак @ индицирует готовность микроЭВМ к выполнению пультовой команды. Жирным шрифтом выделены символы, вводимые оператором. Изменить состояние этой ячейки можно путем задания с клавиатуры нового содержимого, после чего нажимают клавишу «Возврат каретки» <ВК>, и микроЭВМ переходит к обработке микропрограммы загрузки выбранной ячейки памяти новым значением. Например, после выполнения пультовых команд

@ 200/177777 000100 <ВК>

новое значение ячейки 200 станет равным 000100.

Совмещение пульта управления с клавиатурой и устройством отображения дает следующие преимущества: расширение функциональных возможностей пульта управления без существенной переделки микроЭВМ (достаточно написать новые микропрограммы и поместить их в ПЗУ);

повышение технологичности, надежности и компактности микроЭВМ за счет уменьшения количества электрических связей.

В качестве примера устройства связи с пользователем рассмотрим выпускаемый отечественной промышленностью алфавитно-цифровой дисплей «Электроника 15ИЭ-00-013». Дисплей состоит из монитора, блока логики, клавиатуры и источника питания. Он обеспечивает работу в автономном режиме (ввод и редактиро-

вание изображения) и в режиме связи с микроЭВМ (ввод информации с клавиатуры, передача данных в ЭВМ и прием управляющих сигналов). Обмен данными с микроЭВМ осуществляется по двухпроводной линии связи в режиме последовательного доступа со скоростью до 9600 бит/с.

Техническая характеристика дисплея 15ИЭ-00-013

Габаритные размеры, мм	460×690×370
Масса, кг	35
Число отображаемых знаков в строке	80
Число знаковых строк:	
полное	28
видимых	25 (24 информационных, 1 служебная)
Алфавит отображаемых символов (русских и латинских букв)	192
Размер поля изображения, мм	150×220
Формат разложения знака	7 точек×8 строк
Межсимвольный интервал	3 точки
Межстрочный интервал	3 ТВ-строки
Время засветки растровой точки, нс	65
Коэффициент обратного хода по строке	0,2
Частота регенерации, Гц	50
Время непрерывной работы, ч	Не менее 24
Потребляемая электрическая мощность, кВт	Не более 0,25

Совершенствование современных УСП идет по пути улучшения их эргономических характеристик, удобства работы пользователя, экономии времени оператора на формирование и ввод запроса в микроЭВМ, восприятие, мысленную обработку сообщения системы и принятие решения. Для этого создан целый ряд интерактивных устройств ввода, использующих перспективные формы и методы преобразования и передачи информации от человека к ЭВМ (клавиатура, световое перо, планшеты, вращающиеся ручки для перемещения маркера, рычаги и др.).

В универсальной алфавитно-цифровой клавиатуре нажатие клавиши вызывает занесение в регистр литеры 8-битового кода и прерывание микроЭВМ. Смысл литеры зависит от того, как она интерпретируется программой микроЭВМ. Программа может накапливать литеры в буфере до тех пор, пока не будет введен код-ограничитель. После этого цепочка литер распознается процессором и выполняется соответствующая программа.

Кнопочное устройство ввода часто реализуется в виде

функциональной клавиатуры, имеющей обычно 16—32 кнопки. Нажатие каждой кнопки генерирует индивидуальный 4- или 5-битовый код, заносимый в регистр кнопки. После этого следует прерывание текущей программы микроЭВМ и непосредственная передача управления программе, адрес которой определяется по принятому коду. В некоторых функциональных клавиатурах в каждую кнопку встраивается лампочка, управляемая микроЭВМ.

Световое перо является устройством выбора тех или иных элементов, отображаемых на экране монитора. Перо воспринимает резкую вспышку люминофора, когда электронный луч возбуждает его, т. е. формирует данный элемент изображения. Сигнал от светового пера поступает в дисплей и вызывает фиксацию координат электронного пучка в соответствующих регистрах. Затем содержимое регистров передается в процессор микроЭВМ для указания местонахождения отмеченного элемента. Под управлением соответствующей программы в зависимости от функций, выполняемых микроЭВМ, элемент может быть удален, перемещен в другое место экрана, видоизменен и т. п.

Планшет обычно используется для ввода координат элементов и представляет собой систему горизонтальных и вертикальных токопроводящих элементов. Выбор координаты осуществляется прикосновением к соответствующей точке специальным стержнем. Для большинства планшетов положение стержня может быть определено, если от него до поверхности планшета не более 12 мм.

Вращающиеся ручки, потенциометры и рычаги являются устройствами, опрашиваемыми процессором и выдающими аналоговые сигналы, которые после аналого-цифрового преобразования являются исходной информацией для выполнения программы.

6.4. Внешние запоминающие устройства

Общие сведения. В первых микроЭВМ в качестве внешней памяти использовались ленточные перфоленты и фотосчитыватели. Информация на бумажной или майларовой ленте кодировалась в виде совокупности пробивок (например, до 8 отверстий в ряду). При считывании информации перфолента пропускалась между парами светодиод-фотодиод таким образом, что

световой поток от светодиода, проходя через отверстие, вызывал увеличение тока фотодиода. Это соответствует, например, считыванию логической 1. В противоположном случае — логического 0. Вычислительный процесс протекал примерно так: оператор вводил требуемую перфоленку с исходными данными и программами, запускал микроЭВМ на выполнение, и, если процессору требовалась регистрация или ввод промежуточной информации, организовывался вывод данных на перфоленку или выдавалось сообщение о вводе новой перфоленки.

Неудобства в обращении и низкая достоверность записи и воспроизведения информации, малая скорость записи и считывания (до 180 и 1500 байт/с), а также высокий уровень шума привели к необходимости использования в микроЭВМ магнитных накопителей информации. В магнитных накопителях используется свойство магнитных материалов сохранять остаточную намагниченность после прекращения действия вызвавшего ее магнитного поля.

Информация на *магнитный носитель* записывается посредством создания на нем магнитных неоднородностей, определенным образом отображающих записываемую информацию. Для этого на среду, обладающую магнитными свойствами (рабочий слой носителя), воздействуют внешним магнитным полем, под влиянием которого происходит изменение первоначального состояния носителя.

Основным элементом, осуществляющим запись и воспроизведение информации с магнитного носителя, является *магнитная головка* (МГ). Магнитная головка представляет собой незамкнутый магнитопровод (рис. 6.13) из магнитомягкого материала, на который помещается обмотка. Поле записи, воздействующее на магнитный носитель, является полем рассеивания, создаваемым у рабочего зазора. Воспроизведение информации сводится к преобразованию остаточного поля магнитного отпечатка в электрический сигнал. При движении магнитных отпечатков носителя относительно МГ часть остаточного магнитного потока головки замыкается через ее сердечник, в результате обеспечивается магнитная связь между отпечатком и обмоткой сердечника МГ. Эта связь и определяет кодирование считываемой информации.

Из магнитных носителей в микроЭВМ применяются кассетные магнитные ленты (КМЛ) и гибкие (ГМД) и жесткие (МД) магнитные диски.

Кассеты удобны в обращении, обладают большой емкостью и характеризуются низкой стоимостью хранения бита информации. Запись осуществляется на двух или четырех дорожках методами фазового кодирования (логические 0 и 1 кодируются разной полярностью магнитного

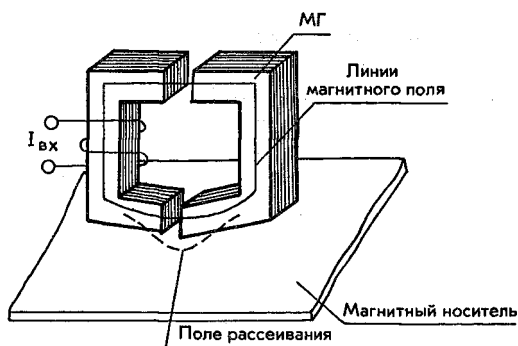


Рис. 6.13

потока). Длина ленты 25—120 м, плотность записи 1—63 бит/мм, рабочие скорости движения 9—75 см/с. Запись на ленту осуществляется в последовательном коде.

Устройство внешней памяти на КМЛ СМ5211. Устройство состоит из двух накопителей на КМЛ типа СМ5204 и работает под управлением микропроцессора КР580ИК80А (рис. 6.14).

Взаимодействие МП с функциональными узлами осуществляется через шину адреса, данных, управления по микропрограммам, записанным в ПЗУ. В регистр управления помещается код управления работой устройства, регистр флажков служит для передачи в МП состояния устройства, а регистры накопителей, режимов накопителей и состояния накопителей используются для связи с КНМЛ.

Команды внешнего управления и информация, предназначенная для записи на КМЛ, записываются в регистр приема. При этом регистр сдвига используется для преобразования параллельного кода в последовательный при записи и воспроизведении данных. Запись и воспроизведение информации на КМЛ осуществляются сигналами шифратора и дешифратора фазовой модуляции сигналов. Байты состояния и уточненного состояния

выдаются через регистр передачи. Таймер совместно с логикой и регистром циклического контроля осуществляет контроль информации на КМЛ, организует подсчет временных интервалов или длины КМЛ.

Устройство выполняет следующие функции: защиту записанной на КМЛ информации; автоматическую за-

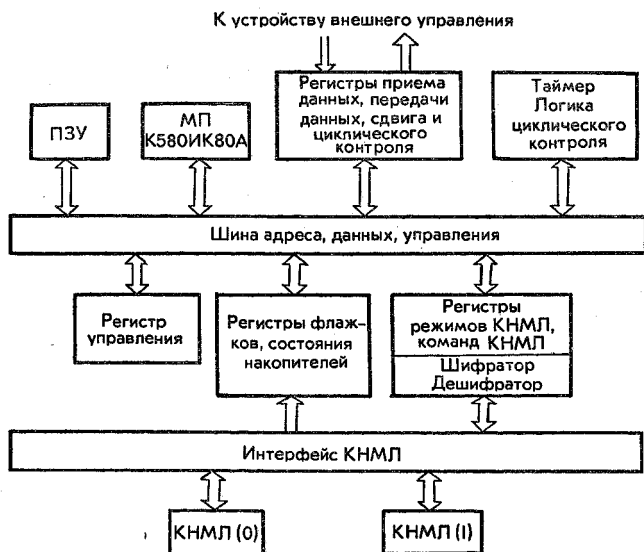


Рис. 6.14

грузку КМЛ в выбранный КНМЛ; выполнение 8 команд внешнего управления от микроЭВМ или другого устройства; контрольное чтение при записи; циклический контроль информации при чтении; автоматическое стирание дефектных участков КМЛ; ускоренный поиск файлов (о файлах см. гл. 7); выдачу информации о состоянии в виде байтов состояния и уточненного состояния. К командам внешнего управления относятся запись метки файла, запись блока данных, чтение блока данных, возврат на файл, возврат на блок, пропуск файла, пропуск блока, перемотка.

Формат информации на КМЛ (рис. 6.15) представляет собой последовательность помеченных записей (файлов), состоящих из блоков длиной 4—256 байт, разделенных межблочными промежутками. Содержимым первого байта метки файла или блока данных является

код предисловия — 10101010. Содержимым последнего байта является код послесловия — 10101010. При записи блока данных перед кодом послесловия записываются 2 байта циклического контроля. При записи метки файла между предисловием и послесловием записываются

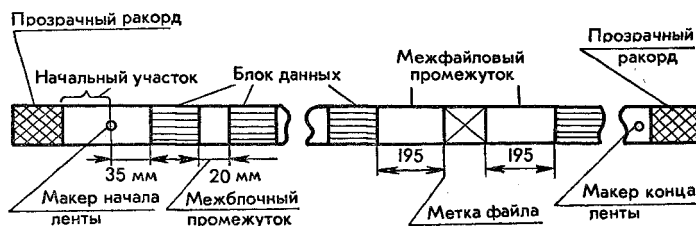


Рис. 6.15

2 байта, содержащие все нули. Младший разряд байта записывается на ленту первым.

Техническая характеристика УВП СМ5211

Габариты, мм	177×482×687
Масса, кг	30
Режим работы	Непрерывный
Потребляемая мощность, В · А	Не более 100
Количество дорожек	2
Номинальная плотность записи, бит/мм	32
Максимальная емкость кассеты, К байт	500
Скорость обмена информацией, К байт/с	1,25
Скорость МЛ в режиме запись/воспроизведение, м/с	0,32
Скорость МЛ в режиме поиска файлов, м/с	1,5
Скорость МЛ в режиме перемотки, м/с	2
Наработка на сбой, бит	10 ⁷

Гибкий магнитный диск. Представляет собой покрытый ферролаком майларовый диск диаметром 130—200 мм, постоянно находящийся в пластиковом чехле (см. рис. 6.16). Привод зажимает центр диска и вращает диск внутри чехла. Прорезь в чехле обеспечивает доступ головки записи-чтения к концентрическим дорожкам. Второе отверстие в чехле обеспечивает оптическое восприятие индексного отверстия на диске, которое отмечает начальную точку каждой дорожки. Емкость диска 1—8 Мбит, скорость передачи информации — 0,25 Мбит/с. Конструктивно НГМД выполняют обычно в виде встраиваемого блока.

Накопитель на гибком магнитном диске ЕС-5088М1.

Схема НГМД дана на рис. 6.16. При установке дискеты в накопитель гибкий магнитный диск (ГМД) центрируется специальной конической прижимающей шайбой, и по сигналу СТАРТ электродвигатель постоянного тока начинает вращать диск. После набора заданного числа обо-

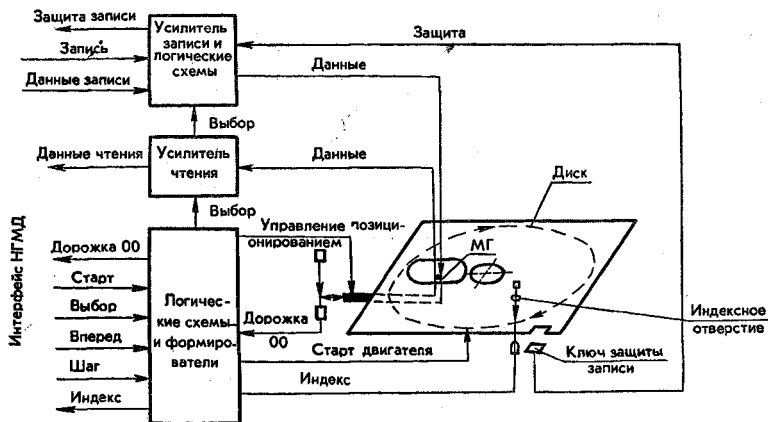


Рис. 6.16

ротов (через промежуток времени менее 1 с) должен быть выполнен возврат МГ к дорожке с номером 00. Для этого необходимо на вход ШАГ подавать импульсы до тех пор, пока устройство позиционирования не сформирует сигнал ДОРОЖКА 00 от фотодатчика. Импульс ШАГ формирует сигналы управления шаговым электродвигателем назад (к дорожкам с меньшим индексом) при низком уровне сигнала ВПЕРЕД на одну дорожку.

МГ со стеклокерамической поверхностью находится в непосредственном контакте с ГМД. Для осуществления лучшего контакта между головкой и ГМД во время обмена данными ГМД дополнительно прижимается к головке. Импульс ИНДЕКС вырабатывается в тот момент, когда индексное отверстие диска проходит под фототранзистором. Сигнал ЗАЩИТА ЗАПИСИ формируется специальным микропереключателем, если в чехле ГМД отсутствует специальный вырез (ключ). В этом случае блокируется интерфейсный сигнал ЗАПИСЬ и разрешается только считывание информации с ГМД.

Запись и считывание данных производится методом частотной модуляции, при котором логические 0 и 1

кодируются разным числом изменений направления магнитного потока за одинаковые промежутки времени (рис. 6.17). Расположение информации на дорожках выполняется в виде неформатных или форматных записей. В последних запись сведений о формате массива предполагается вместе с рабочей информацией.

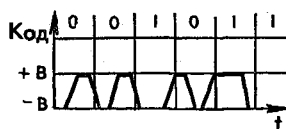


Рис. 6.17

Все операции НГМД выполняет только при активном (низком) уровне сигнала ВЫБОР.

Техническая характеристика НГМД 5088M1

Габариты, мм	205×150×86
Масса, кг	1,4
Частота вращения диска, об/мин	300
Число рабочих поверхностей	1
Емкость, К байт	250
Число дорожек	40 (00—39)
Линейная плотность записи на дорожке	
39, бит/мм	204
Питание от внешних источников	+5 В; +12 В
Параметры периферийных линий:	
$U_{\text{вых}}^0$ В	0—0,4
$I_{\text{вых}}^0$ мА	48
$U_{\text{вых}}^1$ В	2,5—5,0
$I_{\text{вых}}^1$ мА	0,25
Наработка на сбой, бит	10^8
Скорость обмена информацией, К бит/с	250
Диаметр ГМД, мм	130
Габариты пластмассового конверта, мм	130,4×130,4

Накопитель на гибком магнитном диске TEAC FD-55FV-03. НГМД представляет собой ВЗУ емкостью до 1 М байт. В качестве носителя информации используются 133-миллиметровые (5,25 дюйма) гибкие дискеты.

Техническая характеристика НГМД TEAC FD-55FV-03

Габариты, мм	203×146×41,3
Масса, кг	1,5
Частота вращения диска, об/мин	300
Число рабочих поверхностей	2 (стороны 0 и 1)
Максимальная емкость, К байт:	
неформатная запись	1000
форматная запись	655,36

Число дорожек	160 (80×2)
Линейная плотность записи на дорожке	
79, бит/дюйм	5922 (сторона 1)
Наработка на сбой, бит	10 ⁹
Скорость обмена информацией, К бит/с .	250

Накопитель на магнитных дисках. НМД с несменяемым носителем (типа «Винчестер») в отличие от НГМД состоит из нескольких дисков. Для записи данных используется каждая поверхность диска. Данные записываются однотипными блоками (секторами) на концентрических дорожках, которые имеют нумерацию, начиная с внешней (нулевой) дорожки. Благодаря большому числу одновременно доступных дисков, герметичности обеспечивается общая емкость НМД 5—300 М байт.

Перспективы развития ВЗУ. В настоящее время ведутся интенсивные работы по созданию внешних ЗУ, использующих новые принципы организации и технической реализации ЗУ на цилиндрических магнитных доменах (ЦМД) и приборах с зарядовой связью (ПЗС).

ЗУ на цилиндрических магнитных доменах выполняют на основе монокристаллических пленок феррогранатов. В пленках создаются области спонтанной намагниченности (домены), которые сохраняют свои свойства при снятии внешнего поля. Размер кристалла обычно 10×10 мм с информационной емкостью в десятки миллионов бит.

Приборы с зарядовой связью хранят информацию в виде зарядов в потенциальных ямах. Запись и считывание данных происходят путем последовательной передачи зарядов, подобно регистру сдвига. При отключении питания заряды исчезают. Быстродействие ЗУ на ПЗС примерно на порядок выше, чем на ЦМД.

6.5. Устройства ввода и регистрации

Устройства ввода. Важным этапом решения задачи на ЭВМ является ввод исходных данных и регистрация (документирование) результатов в алфавитно-цифровой или графической форме.

В качестве устройств автоматического ввода применяются считыватели с перфолент, перфокарт, магнитные накопители, читающие автоматы. К ручным и полуматематическим устройствам относятся УСП, рассмотренные в § 6.3 и предназначенные для ввода информации в микроЭВМ.

В устройствах ввода алфавитно-цифровых данных каждый символ кодируется определенным образом, например кодом КОИ-7 по ГОСТ 13052—74 (см. табл. 2.3). После этого специальная программа преобразует цепочки введенных в ОЗУ микроЭВМ кодов во внутренний формат центрального процессора. Цифровые данные пере-

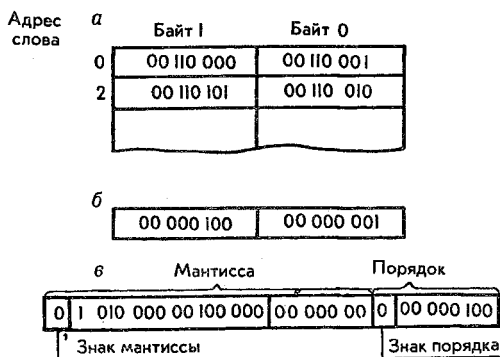


Рис. 6.18

водятся в формат с фиксированной или плавающей запятой, в зависимости от описания в использующей их программе. Символьные константы, как правило, сохраняются в том же виде. Пусть, например, в ОЗУ введены цифры 1, 0, 2, 5, что соответствует кодам (рис. 6.18, а). Тогда после перевода их в 16-разрядное число с фиксированной запятой получим код (рис. 6.18, б). Если указанная цепочка цифр интерпретируется как дробное число 10, 25, то осуществляется перевод в формат с плавающей запятой, и число в этом формате выглядит так, как показано на рис. 6.18, в.

Графическая форма представления информации (графики, рисунки, картографические документы и др.) является самой наглядной, обеспечивает быстрое восприятие ее человеком. Недостаток ее — большое количество двоичных кодов, используемых для описания графического документа. Ввод графического документа в ЭВМ представляет собой сложную техническую задачу, в которой выделяют три стадии: 1) преобразование графической информации в электрический аналоговый сигнал (для этой цели широко используются оптико-электрические способы); 2) преобразование аналогового сигнала

в совокупность цифровых кодов; 3) передача цифровых кодов в ЭВМ.

Методы преобразования графической информации классифицируют по виду траектории перемещения сканирующего пятна (растр-элемента) на следящий и развертывающий.

В устройствах, построенных на основе следящего метода, траектория растр-элемента (рис. 6.19, а) прак-

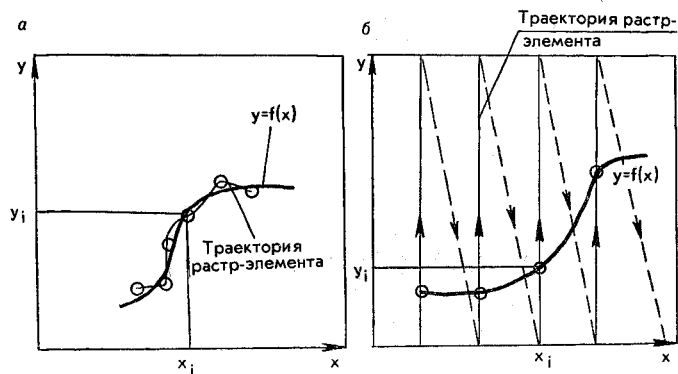


Рис. 6.19

тически совпадает с линией изображения. Этот метод удобен при работе с графическими документами, имеющими дополнительную информацию (пометки, надписи, знаки и пр.), не предназначенную для ввода в ЭВМ. К недостаткам данного метода следует отнести трудность воспроизведения разрывных, пересекающихся или выполненных тонкими линиями изображений.

Развертывающий метод (рис. 6.19, б) предполагает заранее заданную траекторию растр-элемента, параллельную одной из осей координат. Для реализации метода используются более быстродействующие устройства, однако ряд преимуществ его по сравнению со следящим методом (независимость характеристик устройств от сложности и насыщенности графического изображения, высокая точность преобразования и простота построения развертывающей системы) определил в настоящее время доминирующее положение развертывающего метода.

Устройства регистрации (УР). По виду регистрируемой информации УР делятся на печатающие (ПЧУ) и графические регистрирующие (ГРУ) устройства.

Печатающие устройства осуществляют вывод данных преимущественно в алфавитно-цифровом виде, хотя последние модели ПЧУ обладают возможностью печати и графической информации. Поэтому граница между ПЧУ и ГРУ весьма условна и в дальнейшем, вероятно, будет все менее заметна.

Печатающие устройства принято классифицировать по способу регистрации (механическому, физическому или химическому процессу получения визуального изображения) и методу формирования контуров изображения (последовательности действий регистрирующего узла). Существуют механические и немеханические способы регистрации.

В устройствах с *механическим способом* регистрация данных осуществляется путем нанесения красящего вещества на бумагу (при помощи соприкосновения красящего материала или струи красящего вещества). К механическим относятся также перфорационные способы вывода информации.

Немеханические способы регистрации основаны на изменении состояния вещества носителя при воздействии температуры, электрического поля или нанесении на бумагу красящего вещества электромагнитным полем. Хотя немеханические способы сложнее и дороже механических и предполагают в некоторых случаях использование специального дорогостоящего носителя, однако они отличаются высокими быстродействием и надежностью и применяются в высокопроизводительных регистрирующих системах.

В большинстве микропроцессорных систем используются механические способы со знакопечатающим и знаковосинтезирующим (мозаичным) методами формирования контура знака.

Знакопечатающие методы предполагают выбор готового элемента из заранее заданного алфавита (см. § 6.3). На основе этих методов реализованы устройства «Консул-260», «Зоемтрон-520», имеющие быстродействие до 7 ударов/с; модели 100 и 200 фирмы SIEMENS, ADC2100, в которых за счет снижения массы подвижного механизма печати достигается скорость 13—22 знаков/с.

Представляют интерес ПЧУ с лепестковым (дисковым) знаковосителем (рис. 6.20). Знаковоситель состоит из диска с радиально расположенными лепестками. На концах лепестков закреплены литеры. Для печати нужного знака диск поворачивается на требуемый угол

при помощи шагового двигателя, и ударом молоточка (на рисунке не показан) через красящую ленту контур знака наносится на бумагу. Знаконоситель выполнен из легкой прочной пластмассы, что обеспечивает скорость печати до 40 знаков/с. Печать выполняется при прямом и обратном ходе каретки, так как в составе ПЧУ имеется

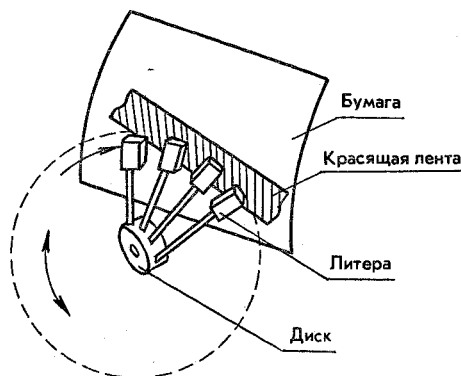


Рис. 6.20

встроенное буферное ЗУ. Алфавит знаков составляет 96 букв и цифр. Механизм используется в ПЧУ СМ6312.

В устройствах *знакосинтезирующего типа* контуры символов формируются из отдельных элементов. Поле символа разбивается на отдельные элементы в виде матрицы, называемой матрицей разложения знака (подобно матрице растрового дисплея, описанного в § 6.3). Элемент формируется путем удара стержня (пуансона, иглы) по бумаге через красящую ленту. Обычно матрица разложения содержит 5×7 , 7×10 и другие варианты элементов (соответственно пуансонов).

В качестве примера рассмотрим мозаичное печатающее устройство СМ6329. ПЧУ позволяет выводить символьную и графическую информацию и состоит из плат управляющей и силовой электроники, узла электропитания, печатающей каретки с плавной направляющей, привода печатающей каретки с шаговым двигателем, механизмов красящей ленты и подачи бумаги.

ПЧУ работает под управлением центрального процессорного элемента ЦПЭ (рис. 6.21). Через порты параллельного (ПРВВ) или последовательного (ПВВ) ввода-вывода ЦПЭ принимает из ЭВМ 8-разрядный код выво-

Данные

ЦПЭ

ПРВВ

Каретка

Подача бумаги

ПРВВ

Пульт

Управление головкой

ПЗУ 10К байт

ОЗУ 2К байт

Адрес, управление

ПРВВ

ПРВВ

К ЭВМ

Печатающая головка

Верхняя игла

Нижняя игла

может задать необходимые режимы работы ЦПЭ. ОЗУ служит для буферного хранения выводимых данных, а в ППЗУ наряду с микропрограммами помещены матрицы разложения знаков.

Матрица разложения знака, точек	9×9
Алфавит символов	96
Максимальное число символов:	
на дюйм	17
в строке	233
Минимальное число символов (широкий шрифт):	
на дюйм	5
в строке	68
Плотность вывода графической информации, точек/дюйм:	
по вертикали	72
по горизонтали	240
Число типов шрифта	6
Задание типа шрифта Программное или с пульта	
Число печатающих игл	9

Расположение игл	В двух колонках — 4 и 5 в шахматном порядке
Шаг между иглами, мм	0,35
Диаметр иглы, мм	0,35
Максимальная частота срабатывания иглы, ударов/с	900
То же при длительной нагрузке	200

Графические регистрирующие устройства (графопостроители), подобно читающим автоматам, по способу перемещения регистрирующего узла делятся на два больших класса: с *произвольным* и *растровым сканированием*.

По способу перемещения носителя в момент регистрации различают ГРУ планшетного и барабанного типов.

Планшетные графопостроители выполняют регистрацию данных на неподвижном листе бумаги, удерживаемом на специальном столе электростатическим полем, вакуумом или путем натяжения. В таких ГРУ регистрирующий узел при помощи шаговых двигателей, системы тросов и направляющих перемещается по осям *x* и *y* на заданное число шагов (дискретных составляющих). При этом перо, установленное на регистрирующем узле, вычерчивает заданную траекторию. Размеры планшетных ГРУ колеблются от 30×45 см до 2×3 м. В некоторых случаях в качестве пера используется источник света для экспонирования фотографических материалов или резец для снятия слоя материала.

Графопостроители барабанного типа вычерчивают изображение на свернутой в рулон бумаге, движущейся относительно каретки ГРУ. Регистрирующий узел совершает возвратно-поступательное движение перпендикулярно к направлению бумаги. Натяжение бумаги обеспечивается подающим и приемным роликами, вращающимися в прямом и обратном направлениях.

Графопостроители выполняют следующие команды: опустить перо, поднять перо, сменить перо, сместиться на один шаг влево (вправо), сместиться на один шаг вверх (вниз). Графопостроители со встроенным микропроцессором могут выполнять более сложные команды: вычерчивание окружности, дуги, вектора, знака и др. В этом случае непосредственное управление исполнительными механизмами выполняет микропроцессор, а для управляющей ЭВМ графопостроитель является цифровым блоком с заданной системой команд. Каждая команда инициирует выполнение определенной последовательности действий ГРУ.

Техническая характеристика графопостроителя ЭМ7042А

Габаритные размеры, мм	665×550×238
Масса, кг	28
Размеры рабочего поля, мм	420×300
Максимальная погрешность, мм	0,2
Разрешающая способность, мм	Не более 0,025
Максимальная скорость движения, мм/с	400
Максимальное ускорение, м/с ²	1,6
Время непрерывной работы, ч	Не менее 16
Максимальная потребляемая мощность, кВт	0,25
Время наработки на отказ, ч	300

Контрольные вопросы

1. Какие группы периферийных устройств вам известны?
2. Для чего предназначены эти периферийные устройства?
3. Какое периферийное оборудование наиболее удобно для ведения диалога человек — ЭВМ?
4. Какое периферийное оборудование необходимо для управления технологическим процессом?
5. Чем характеризуются современные внешние запоминающие устройства микроЭВМ?
6. Сравните по быстродействию и информационной емкости известные вам типы ВЗУ.
7. Какие физические принципы используются для построения устройств ввода и регистрации?
8. Какие типы устройств регистрации вы знаете? Чем они характеризуются?
9. Можно ли при помощи модуля 15КА-60/4-009 сформировать гармонический сигнал с амплитудой 10 В и частотой 1000 Гц?
10. Можно ли при помощи модуля 15КА-60/8-010 построить 16-канальное УСО, если датчики имеют диапазон работы 1 В, а частотный диапазон принимаемых сигналов — 10 кГц?
11. Какое максимальное число телевизионных строк можно получить на экране телевизионного растрового устройства отображения, если $T_c = 64$ мкс?
12. Как на экране дисплея могут быть представлены знаки «I», «J», «a», «B» в формате 8×8 точек?
13. Какое максимальное число знаков можно одновременно отображать на экране дисплея «Электроника 15ИЭ-00-013»?
14. В какие моменты времени осуществляется обмен информацией между микроЭВМ и дисплеем в режиме независимого, прямого и последовательного доступов?
15. Какие вы знаете современные интерактивные устройства ввода?
16. Как выполняется ввод данных и управление микроЭВМ при помощи универсальной алфавитно-цифровой клавиатуры?
17. Почему емкость ГМД с неформатной записью выше, чем с форматной?
18. Чем различаются знакопечатающие и знаковинтезирующие методы регистрации данных?
19. Какой механизм регистрации данных используется в ПЧУ СМ6312?
20. Какой метод формирования контура изображения применяется в ПЧУ СМ6329?

21. Какая максимальная толщина линии может получиться при многократном повторении траектории пера графопостроителя ЭМ7042А, если диаметр пера — 1 мм?

Упражнения

1. Начертите структурную схему УСО для микроЭВМ, принимающей информацию с датчика. Диапазон работы датчика 1 мВ, частотный диапазон входного сигнала — 2000 Гц.

2. Начертите структурную схему УСО для микроЭВМ, формирующей гармонический сигнал частотой до 2000 Гц, амплитудой 1 В.

3. Начертите структурную схему 4-канального УСО с одним АЦП. Диапазон работы датчиков — 10 мВ, частотный диапазон исследуемых сигналов — 100 Гц.

4. Рассчитайте минимальную тактовую частоту генератора прямоугольных импульсов в устройстве отображения с $T_c = 64$ мкс и числом точек в строке 1024.

5. Выберите тип БИС ОЗУ для построения ОЗУ регенерации алфавитно-цифрового дисплея с 80 знаками в строке, 24-знаковыми строками, разложением знака в формате 5×7 ; межсимвольный интервал — три точки, межстрочный интервал — одна строчка.

6. Опишите последовательность действий при записи в первый регистр микроЭВМ «Электроника-60» кода 2000 (восьмеричная система счисления).

7. Рассчитайте число букв текста, которое можно поместить на 10 м кассетной магнитной ленты УВП СМ5211, если каждая буква кодируется кодом КОИ-7.

8. Рассчитайте время полного оборота диска НГМД ЕС 5088М1. Сколько информации (в битах) сможет передать НГМД за это время?

9. Рассчитайте, за какое время будет выведен текст из 60 букв, если в качестве регистрирующего устройства используется ПЧУ СМ6329.

10. Рассчитайте, за какое время будет вычерчен квадрат со стороной 20 см, если в качестве регистрирующего устройства используется графопостроитель ЭМ7042А.

Глава 7. ОСНОВЫ ПРОГРАММИРОВАНИЯ МИКРОЭВМ

7.1. Программное обеспечение микроЭВМ

Аппаратное обеспечение микроЭВМ — это универсальный инструмент, пригодный для решения широкого класса задач. Однако конкретная задача может быть решена только в случае, если вычислительной машине известен алгоритм — точное предписание, которое определяет ход вычислительного процесса, ведущего от варьируемых начальных данных к искомому результату.

Запись алгоритма в форме, воспринимаемой ЭВМ, называется программой. На время выполнения программа помещается в оперативное запоминающее устройство или хранится в ПЗУ (ППЗУ).

Совокупность программ и документации на них, предназначенных для реализации целей и задач микроЭВМ, называется программным обеспечением (ПО). Программное обеспечение условно подразделяют на общее (системное) и специальное (функциональное).

Общее ПО — это часть программного обеспечения, которая предназначена для автоматизации разработки программ, организации и контроля вычислительного процесса при решении широкого класса задач. Общее ПО обычно поставляется вместе с микроЭВМ и предполагает использование строго определенного набора технических средств.

Специальное ПО образуют программы, ориентированные на решение конкретной задачи или группы взаимосвязанных задач. Для реализации специального ПО, помимо универсальных, могут использоваться специализированные аппаратные средства, которые наиболее приспособлены для решения поставленной задачи. Специальное ПО создается обычно под управлением и на основе общего ПО. Оно может включать отдельные программные модули и компоненты общего ПО. *Программный модуль* — это программа, рассматриваемая как единое целое при хранении, трансляции и объединении с другими программами.

В структуре ПО можно выделить следующие функциональные части: операционную систему (ОС), пакеты прикладных программ (ППП), пакет программ диагностики и технического обслуживания (ППДТО) (рис. 7.1).

Операционной системой называется комплекс программ, предназначенный для планирования и органи-

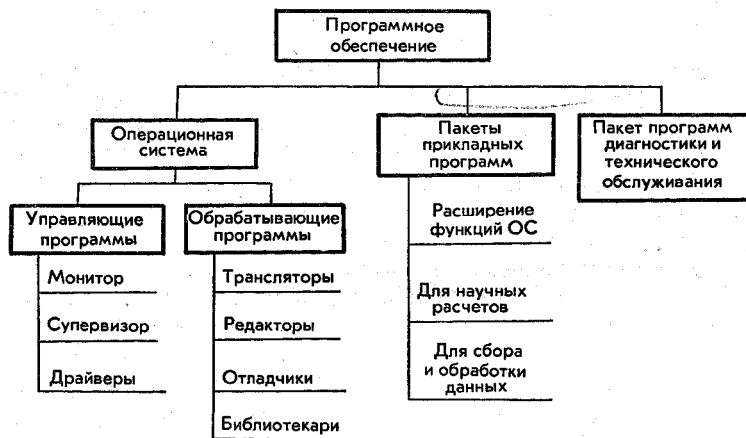


Рис. 7.1

зации процесса обработки, ввода-вывода и управления данными, распределения ресурсов, подготовки и отладки программ и других вспомогательных операций обслуживания. Основными компонентами ОС являются управляющие и обрабатывающие программы.

К *управляющим* относятся программы, на которые возложены функции взаимодействия с оператором, управления выполнением других программ, вводом-выводом данных, распределением памяти и других ресурсов. Группу *обрабатывающих* составляют программы автоматизации подготовки, редактирования и отладки программ.

Совокупность программных модулей, рассчитанных на решение определенного класса задач и оформленных в соответствии с определенными требованиями, называют пакетом прикладных программ. В состав ПО микроЭВМ может входить несколько ППП, каждый из которых является основой того или иного специального ПО.

Программы диагностики и технического

обслуживания, используют для профилактики, диагностики и ремонта микроЭВМ. ППДТО могут функционировать в составе аппаратных средств микроЭВМ или в совокупности с дополнительным контрольно-испытательным и тестовым оборудованием.

7.2. Система команд

Общие сведения. Одной из основных характеристик микроЭВМ, определяющей возможность использования готовых программных модулей, является система команд — полная совокупность команд, которые может выполнять ЭВМ.

Команда представляет собой двоичный код, определяющий операцию, производимую вычислительной машиной, и данные, участвующие в операции (так называемые операнды). Команда содержит операционную и адресную части (рис. 7.2, а). В операционной части указывается код операции (КОП) — вид машинной операции (сложение, вычитание, умножение, пересылка данных и т. п.). В адресной части задается информация о местонахождении операндов. Если ЭВМ выполняет M различных операций, то число двоичных разрядов КОП $N_{\text{коп}} = \lceil \log_2 M \rceil$, где $\lceil A \rceil$ — ближайшее целое число, большее A . Например, при $M = 200$ $N_{\text{коп}} = 8$.

Число разрядов адресной части определяется количеством операндов, участвующих в операции, и количеством разрядов адреса операнда, зависящим от объема оперативной памяти (V_n), измеряемого в байтах. Различают *безадресные* (рис. 7.2, б), *одноадресные* (рис. 7.2, в) и *двухадресные* (рис. 7.2, г) команды. Соответственно и число двоичных разрядов адресной части $N_0 = 0$; $N_{\text{од}} = \lceil \log_2 V_n \rceil$; $N_{\text{дв}} = 2 \lceil \log_2 V_n \rceil$. Например, при сравнительно небольшом $V_n = 64$ К байт $N_{\text{дв}} = 32$ бит. Полная длина двухадресной команды в примере составит $N_k = N_{\text{коп}} + N_{\text{дв}} = 40$ бит. Такой размер команды неудобен для реализации и плохо согласуется с форматами данных, выбираемыми из ряда 8, 16, 32, 64 бит. Поэтому и *формат команды*, т. е. ее структуру, позволяющую распознавать составные части, стремятся привести в соответствие с форматом данных. Для этой цели в микроЭВМ используют неявные методы адресации и переменную длину команды.

В настоящее время среди микропроцессоров и микроЭВМ наиболее распространенными являются две систе-

мы команд: DEC (разработана в США фирмой Digital Equipment Corporation для микроЭВМ PDP-11, LSI-11) и Intel (для МП Intel 8080). Многие выпускаемые в СССР МП и микроЭВМ имеют эти системы команд: DEC — микроЭВМ «Электроника-60», ДВК, МП К1801ВМ1 (ВМ2, ВМ3), МПК К588, мини-ЭВМ СМ-3, СМ-4, быто-

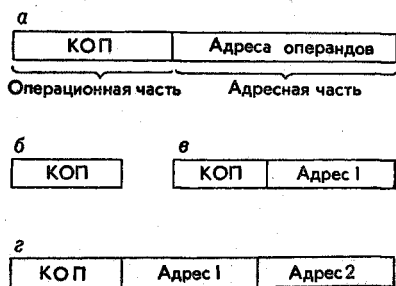


Рис. 7.2

вой компьютер БК-0010 и др.; Intel — мини-ЭВМ СМ-1800, МП КР580ВМ80, бытовые компьютеры «Микроша», РК-86. Более поздней версией системы команд МП Intel 8080 стала система команд МП Intel 8086. На этом МП построены персональные компьютеры типа IBM PC (в СССР ЕС 1840, ЕС 1841, Искра-1030 и др.).

Система команд DEC. В основу построения системы команд DEC положена концепция, в соответствии с которой процессор обращается к регистрам внешних устройств как к ячейкам памяти. Это упрощает логическую структуру микроЭВМ, поскольку с точки зрения программиста вычислительная среда состоит из восьми 16-разрядных регистров общего назначения (РОН), процессора и памяти объемом 64 К байт (32 К 16-разрядных слов), отдельные области и ячейки которой физически могут принадлежать совершенно независимым устройствам.

Весь набор команд принято разделять в соответствии с их функциональным назначением на пять групп: 1) команды пересылки данных; 2) арифметические команды; 3) логические команды; 4) команды ветвления; 5) команды управления. При этом операции могут выполняться как со словами, так и с байтами.

Команды пересылки служат для передачи информации между регистрами процессора, ячейками

памяти или регистром и ячейкой памяти. Мнемоническое* обозначение операции — MOV.

Арифметические и логические команды предназначены для выполнения арифметических и логических действий с одним или несколькими операндами. Команда пересылки имеет двухадресный формат. Форма-

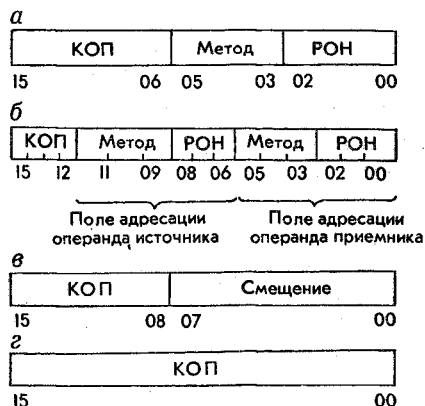


Рис. 7.3

ты арифметических и логических команд могут быть как двухадресными (рис. 7.3, *а*), так и одноадресными (рис. 7.3, *б*).

В формате одноадресной команды разряды с номерами 15—06 содержат код операции, который определяет выполняемую команду. Разряды 05—00 образуют шестиразрядное поле, которое именуется полем адресации операнда приемника и состоит из двух подполей:

а) разряды 02—00 определяют один из восьми РОН, который использует данная команда;

б) разряды 05—03 определяют способ использования выбранного регистра (метод адресации), причем разрядом 05 определяется прямой или косвенный метод адресации. Состав одноадресных команд приведен в табл. 7.1.

Операции над двумя операндами (такие, как сложение, пересылка, сравнение) выполняются с помощью команд, в которых имеются два поля адреса. Задание разрядов в поле адресации первого и второго операндов определяет различные методы и регистры общего назначения.

* Мнемоника — совокупность приемов и способов, облегчающих запоминание.

Таблица 7.1

Мнемоника	Команда	Код	Описание
1	2	3	4
CLR(B)	Очистка	*050DD	$(D) = 0$ $N = V = C = 0$, $Z = 1$
COM(B)	Инвертирование	*051DD	$(D) = \text{«NOT» } (D)$ $N = 1$, если $(D) < 0$ $V = 0$, $C = 1$ $Z = 1$, если $(D) = 0$
INC(B)	Прибавление единицы	*052DD	$(D) = (D) + 1$ $N = 1$, если $(D) < 0$ $Z = 1$, если $(D) = 0$ $V = 1$, если $(D) = 077777$ C — не изменяется
DEC(B)	Вычитание единицы	*053DD	$(D) = (D) - 1$ $N = 1$, если $(D) < 0$ $Z = 1$, если $(D) = 0$ $V = 1$, если (D) было равно 100 000 C — не изменяется
NEG(B)	Изменение знака	*054DD	$(D) = -(D)$ $N = 1$, если $(D) < 0$ $Z = 1$, если $(D) = 0$ $V = 1$, если (D) стало равным 100 000 $C = 0$, если $(D) = 0$
TST(B)	Проверка	*057DD	$(D) = (D)$ $N = 1$, если $(D) < 0$ $Z = 1$, если $(D) = 0$ $V = C = 0$
ASR(B)	Арифметический сдвиг вправо	*062DD	(D) -сдвинутое на одну позицию вправо (D) $N = 1$, если старший разряд (D) равен 1 $Z = 1$, если $(D) = 0$ $V = N \text{«XOR» } C$ C — загружается содер- жимым младшего раз- ряда операнда
ASL(B)	Арифметический сдвиг влево	*063DD	(D) -сдвинутое на одну позицию влево (D) $N = 1$, если $(D) < 0$ $Z = 1$, если $(D) = 0$ $V = N \text{«OR» } C$ C — загружается содер- жимым старшего разря- да операнда

1	2	3	4
ROR(B)	Циклический сдвиг вправо	*060DD	(D)-сдвинутое циклически на одну позицию вправо (D) N=1, если (D) < 0 Z=1, если (D) = 0 V=N«XOR»C C-загружается содержимым младшего разряда операнда
ROL(B)	Циклический сдвиг влево	*061DD	(D)-сдвинутое циклически на одну позицию влево (D) N=1, если (D) < 0 Z=1, если (D) = 0 V=N«OR»C C — загружается содержимым старшего разряда операнда
ADC(B)	Прибавление переноса	*055DD	(D) = (D) + (C) N=1, если (D) < 0 Z=1, если (D) = 0 V=1, если перед выполнением операции (D)=077777, а (C)=1
SBC(B)	Вычитание переноса	*056DD	(D) = (D) - (C) N=1, если (D) < 0 Z=1, если (D) = 0 V=1, если перед выполнением команды (D) = =100000 C=1, если перед выполнением операции (D) = =0 и (C)=1
SXT	Расширение знака	0067DD	(D)=0, если N=0 (D) = -1, если N=1 N — не изменяется Z=1, если (N)=0 V=0, C — не изменяется
SWAB	Перестановка байтов	0003DD	Байт 1/Байт 0 = Байт 0/Байт 1 N=1, если старший разряд байта 0 равен 1 Z=1, если младший байт результата равен 0 V=C=0

Окончание табл. 7.1

1	2	3	4
MEPS	Чтение ССП	1067DD	(D) = ССП N=1, если разряд 7 ССП равен 1 Z=1, если разряды 00—07 ССП равны 0 V=0, C — не изменяет- ся
MTPS	Запись ССП	1064SS	ССП = (S) N, Z, V, C устанавли- ваются в соответствии с разрядами 00—03 (S)

Поле адресации операнда источника используется для выборки операнда источника. Поле адресации операнда приемника применяют для выборки операнда приемника и занесения результата. Например, по команде ADD A, B складывается содержимое ячейки A (операнд источника) с содержимым ячейки B (операнд приемника). После выполнения операции сложения в ячейке B будет находиться результат операции, а содержимое ячейки A не изменится. Состав двухадресных команд приведен в табл. 7.2.

Таблица 7.2

Мнемоника	Команда	Код	Описание
1	2	3	4
MOV (B)	Пересылка	*1SSDD	(D) = (S) N=1, если (S) < 0 Z=1, если (S) = 0 V=0, C — не изменя- ется
CMP (B)	Сравнение	*2SSDD	(S) - (D) N=1, если резуль- тат < 0 Z=1, если резуль- тат = 0 V=1, если было ариф- метическое переполне- ние C=1, если был перенос из старшего разряда ре- зультата

Окончание табл. 7.2

1	2	3	4
ADD	Сложение	06SSDD	$(D) = (S) + (D)$ $N=1$, если $(D) < 0$ $Z=1$, если $(D) = 0$ $V=1$, если было арифметическое переполнение $C=1$, если был перенос из старшего разряда результата
SUB	Вычитание	16SSDD	$(D) = (D) - (S)$ $N=1$, если $(D) < 0$ $Z=1$, если $(D) = 0$ $V=1$, если было арифметическое переполнение $C=0$, если был перенос из старшего разряда результата
BIT(B)	Проверка разрядов	*3SSDD	$(S) \llcorner \text{AND} \llcorner (D)$ $N=1$, если старший разряд результата установлен $Z=1$, если все разряды результата равны 0 $V=0$, C — не изменяется
BIC(B)	Очистка разрядов	*4SSDD	$(D) = (\llcorner \text{NOT} \llcorner (S)) \llcorner \text{AND} \llcorner (D)$ $N=1$, если старший разряд (D) равен 1 $Z=1$, если $(D) = 0$ $V=0$, C — не изменяется
BIS(B)	Логическое сложение	*5SSDD	$(D) = (S) \llcorner \text{OR} \llcorner (D)$ $N=1$, если старший разряд (D) равен 1 $Z=1$, если $(D) = 0$ $V=0$, C — не изменяется
XOR	Исключающее ИЛИ	074RDD	$(D) = R \llcorner \text{XOR} \llcorner (D)$ $N=1$, если $(D) < 0$ $Z=1$, если $(D) = 0$ $V=0$, C — не изменяется

В табл. 7.1, 7.2 использованы следующие обозначения: R — регистр общего назначения; SS — поле адреса

ции операнда источника; DD — поле адресации операнда приемника; «NOT», «AND», «OR», «XOR» — логические функции НЕ, И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ; (S) — операнд-источник; (D) — операнд-приемник; ССП — слово состояния процессора; N, Z, V, C — признаки выполнения операции; N — отрицательный результат, Z — рав-

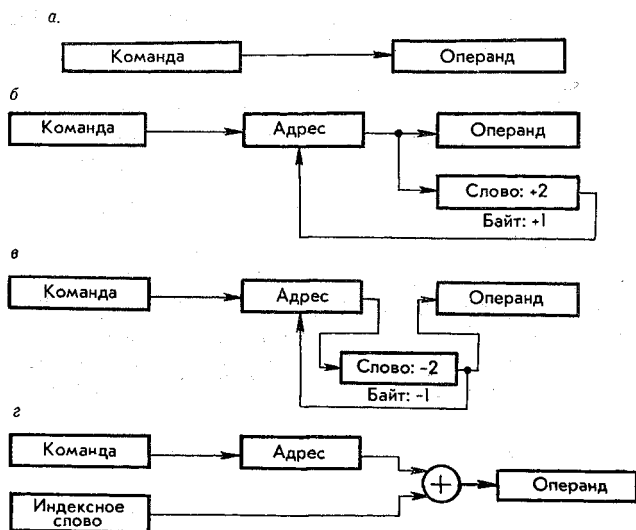


Рис. 7.4

ный нулю, V — переполнение, C — перенос; B — признак байтовой операции; * означает 0 — в командах со словами, 1 — в байтовых командах.

При рассмотрении системы команд используют мнемонические обозначения, которые образованы сокращением английского названия соответствующей операции: MOV — команда пересылки (move — передвинуть, переместить), ADD — команда сложения (add — сложить), SUB — вычитание (subtract — вычесть, отнять), INC — увеличить на единицу (increment — увеличение, приращение) и т. д. После мнемонического обозначения указывается адреса операндов или сами операнды.

Поскольку основные поля команды имеют три двоичных разряда, для записи машинного кода команды принята восьмеричная система счисления.

Методы адресации в системе команд DEC условно разделены на две группы: методы прямой (рис.

7.4) и косвенной адресации (рис. 7.5). К методам *прямой адресации* относятся регистровый, автоинкрементный, автодекрементный и индексный. Эти методы кодируются соответственно восьмеричными цифрами 0, 2, 4 и 6.

Время выполнения команды T_k в общем виде состоит из времени выполнения операции T_v , времени выборки операндов (источника $T_{и}$ и приемника $T_{п}$): $T_k = T_v + T_{и} + T_{п}$ и различно для разных команд. Время T_v зависит от типа операции, а в ряде случаев и от конкретных значений данных, участвующих в операции. Слагаемые $T_{и}$ и $T_{п}$ всецело определяются методом адресации. В табл. 7.3 приводятся типовые значения $T_{и}$ и $T_{п}$ для микроЭВМ «Электроника-60» с циклом работы центрального процессора 400 нс.

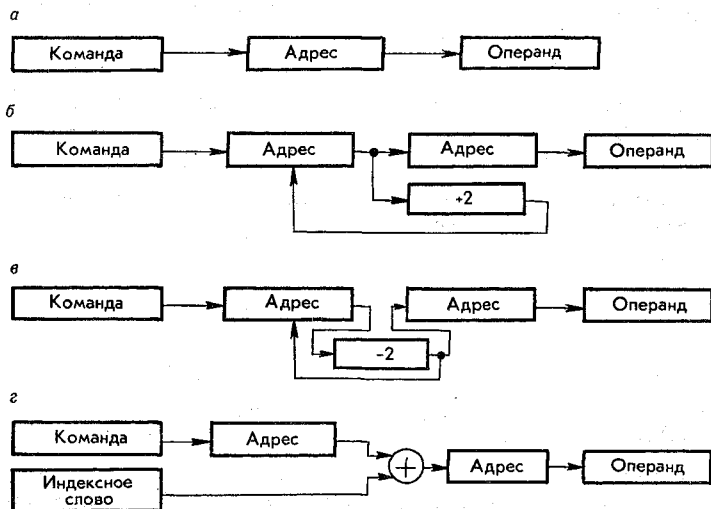


Рис. 7.5

При *регистровом методе адресации* (см. рис. 7.4, а) операнд находится в регистре. Это наиболее оперативный метод адресации, поскольку при его реализации не тратится машинное время на выборку операнда из памяти. Здесь и далее приводятся примеры команд, реализующих различные методы адресации. Дается символическое обозначение команды, используемое для записи ее на языке АССЕМБЛЕР, восьмеричный код, наименование и содержание выполняемого действия, а также содер-

CLR (R5)+

005025

Очистка

Действие: Содержимое ячейки, адрес которой содержится в R5, очищается, после чего адрес (содержимое R5) увеличивается на 2.
До выполнения операции: После выполнения операции:

20000/ 005025 R5/ 030000
30000/ 111116

20000/ 005025 R5/ 030002
30000/ 000000

CLRB (R5)+

105025

Очистка байта

Действие: Очищается выбранный байт, адрес которого содержится в R5, после чего адрес (содержимое R5) увеличивается на единицу.
До выполнения операции: После выполнения операции:

20000/ 105025 R5/ 030000
30000/ 111116

20000/ 105025 R5/ 030001
30000/ 111000

ADD (R2)+,R4

062204

Сложения

Действие: Операнд, адрес которого содержится в R2, складывается с содержимым R4. Результат заносится в R4, а содержимое R2 увеличивается на 2.
До выполнения операции: После выполнения операции:

100/ 012204 R2/ 000204
204/ 002000 R4/ 001000

100/ 012204 R2/ 000206
204/ 002000 R4/ 003000

При автодекрементном методе адресации (рис. 7.4, в) адрес операнда помещается в регистр, перед выборкой операнда содержимое регистра уменьшается на 2 в случае обращения к слову и на 1 — в случае обращения к байту.

Автодекрементный метод также используется для последовательного обращения к элементам массива или стеку без дополнительных команд уменьшения адреса операнда. В системе DEC принята такая организация стека, при которой для записи в стек применяется автодекрементный метод адресации, а для считывания — автоинкрементный. В качестве *указателя стека* — регистра, в котором хранится адрес последней занятой ячейки стека, используется регистр 6. Регистр 7 занят процессором для организации счетчика адреса команд.

Если команды выполняются последовательно, в естественном порядке, то содержимое R7 перед выполнением команды увеличивается на 2, т. е. он подготавливается к выборке очередной команды. При необходимости нарушения естественного порядка команд (ветвления, обхода ненужного участка программы, обращения к подпро-

грамме) в R7 помещается адрес команды, следующей за выполняемой.

DEC -(R0)

005340

Вычитание единицы

Действие: содержимое R0 уменьшается на 2 и используется как исполнительный адрес. Из операнда, выбранного из ячейки по этому адресу, вычитается единица.

До выполнения операции:

После выполнения операции:

100/ 005340 R0/ 017776
017774/ 000011

100/ 005340 R0/ 017774
017774/ 000010

Представляет интерес команда используемая для уменьшения содержимого регистра R0 на 4 с минимальными затратами памяти программ:

CMR -(R0),-(R0)

024040

Сравнение

До выполнения операции:

После выполнения операции:

100/ 024040 R0/ 001004
1002/ 000200

100/ 024040 R0/001000
1002/ 000200

При индексном методе адресации (рис. 7.4, г) адрес операнда определяется как сумма содержимого выбранного РОН с индексным словом, размещенным в следующей за командным словом ячейке. Этот метод адресации используется в основном для произвольного обращения к элементам структуры данных.

NEB 200(R4)

**005464
000200**

**Изменение
знака**

Действие: Адрес операнда определяется прибавлением к содержимому R4 кода 200, после чего в ячейке с вычисленным адресом изменяется знак.

До выполнения операции:

После выполнения операции:

100/ 005464 R4/ 000002
102/ 000200
202/ 000010

100/ 005464 R4/ 000002
102/ 000200
202/ 177770

SUB 30(R2),20(R5)

**166265
000030
000020**

Вычитание

Действие: Содержимое ячейки, адрес которой определяется сложением 30 с кодом, хранящимся в R2 (операнд источника), вычитается из содержимого ячейки, адрес которой определяется сложением

нием 20 с содержанием R5 (операнд приемника). Результат записывается по адресу операнда приемника.

До выполнения операции:

После выполнения операции:

100/ 166265 R2/ 000200
102/ 000030 R5/ 000400
104/ 000020
230/ 001010
420/ 006060

100/ 166265 R2/ 000200
102/ 000030 R5/ 000400
104/ 000020
230/ 001010
420/ 005050

К методам косвенной адресации относятся косвенно-регистровый (код 1), косвенно-автоинкрементный (код 3), косвенно-автодекрементный (код 5) и косвенно-индексный (код 7). Косвенные методы адресации предполагают использование не адреса операнда, а адреса адреса операнда и по сравнению с прямыми методами при нахождении операнда требуют на одно обращение к памяти больше.

Косвенно-регистровый метод адресации (рис. 7.5, а) предполагает размещение адреса операнда непосредственно в регистре.

ASR R5

006215

Арифметический
сдвиг вправо

Действие: содержимое ячейки, адрес которой содержится в R5, сдвигается на один разряд вправо.

До выполнения операции:

После выполнения операции:

100/ 006215 R5/ 001700
1700/ 100007

100/ 006215 R5/ 001700
1700/ 140003

При реализации *косвенно-автоинкрементного метода* (рис. 7.5, б) адрес адреса операнда, размещенный в регистре, увеличивается на 2 после выполнения операции.

ASL R(R2)+

006332

Арифметический
сдвиг влево

До выполнения операции:

После выполнения операции:

1000/ 006332 R2/ 010300
1010/ 000001
10300/ 001010

1000/ 006332 R2/ 010302
1010/ 000002
10300/ 001010

Обращение к операнду по *косвенно-автодекрементному методу* (рис. 7.5, в) означает предварительное уменьшение на 2 размещенного в регистре R адреса адреса операнда.

Косвенно-индексный метод адресации (рис. 7.5, г)

RDR 8-(R0)

006050

Циклический сдвиг
вправо

До выполнения операции:

После выполнения операции:

100/ 006050 R0/ 010776

100/ 006050 R0/ 010774

10100/ 100007

10100/ 040003

10774/ 010100

10774/ 010100

Признак С установлен в 0.

Признак С устанавливается в 1.

позволяет определить адрес операнда путем суммирования содержимого регистра и индексного слова, размещенного в ячейке, следующей за командным словом.

RDL 81000(R2)

006172

Циклический
сдвиг влево

До выполнения операции:

После выполнения операции:

1020/ 006172 R2/ 000100

1020/ 006172 R2/ 000100

1022/ 001000

1022/ 001000

1050/ 000002

1050/ 000003

1100/ 001050

1100/ 001050

Признак С установлен в 1.

Признак С устанавливается в 0.

Особое место занимают методы адресации, использующие в качестве регистра счетчик адреса команд (СК) — R7 (табл. 7.4). Они предоставляют програм-

Таблица 7.4

Двоичный код	Обозначение	Метод адресации	Функция
010	#	Непосредственный	Операнд выбирается из ячейки, следующей за командным словом
011	@#	Абсолютный	Из ячейки, следующей за командным словом, выбирается адрес операнда
110	X(R7)	Относительный	Операнд выбирается из ячейки, адрес которой определяется как сумма содержимого счетчика команд (СК) и ячейки, следующей за командным словом
111	@X(R7)	Косвенно-относительный	Из ячейки, адрес которой определяется как сумма содержимого СК и ячейки, следующей за командным словом, выбирается адрес операнда

Примечание. X — индекс.

мисту дополнительные возможности, упрощающие разработку программы.

Команды ветвления (табл. 7.5, рис. 7.3, в) используются в том случае, если необходимо изменить

Таблица 7.5

Мнемоника	Команда	Код
BR	Ветвление безусловное	000400
BNE	Ветвление, если результат не равен нулю	001000
BEQ	Ветвление, если результат равен нулю	001400
BPL	Ветвление, если результат положительный	100000
BMI	Ветвление, если результат отрицательный	100400
BVC	Ветвление, если нет арифметического переполнения	102000
BVS	Ветвление, если есть арифметическое переполнение	102400
BCC	Ветвление, если нет переноса	103000
BCS	Ветвление, если есть перенос	103400
BGE	Ветвление, если результат больше или равен нулю	002000
BLT	Ветвление, если результат меньше нуля	002400
BGT	Ветвление, если результат больше нуля	003000
BLE	Ветвление, если результат меньше или равен нулю	003400
BHI	Ветвление, если больше	101000
BLOS	Ветвление, если меньше или равно	101100
BHIS	Ветвление, если больше или равно	103000
BLO	Ветвление, если меньше	103400
JMP	Безусловный переход	0001DD
JSR	Обращение к подпрограмме	004RDD
RTS	Возврат из подпрограммы	00020R
MARK	Восстановление указателя стека	0064NN
SOB	Вычитание единицы и ветвление	077RNN

естественную последовательность команд. Команда условного перехода анализирует состояние признака (N, Z, V или C), установленного при выполнении арифметической, логической операции или операции пересылки. Если условие верно (признак установлен или сброшен), то выполняется команда, адрес которой определяется как сумма

$$A = (CK) + 2 + 2X,$$

где (CK) — содержимое CK, указывающее адрес команды ветвления, а X — восьмизначный код смещения (см. рис. 7.3, в). В противном случае выполняется следующая команда.

100/ 000401

102/ 10100

104/ 005001

В результате выполнения команды ветвления, расположенной в ячейке с адресом 100, следующей выполняется команда, расположенная в ячейке с адресом 104.

Использование команды ветвления рассмотрим на примере программы ожидания готовности внешнего устройства.

100/ 105737

102/ 177564

104/ 100375

Команда (100) устанавливает признаки по младшему байту операнда (младший байт соответствует четному адресу). В качестве операнда выступает регистр состояния дисплея — пультового терминала. Если 7-й разряд (начиная с нулевого) в регистре состояния сброшен, то признак N устанавливается в 0, и в результате выполнения команды (104) в СК установится код 100. Программа выполняется до тех пор, пока 7-й разряд регистра состояния не установится. В этом случае после команды (104) выполнится команда (106).

Команды управления (табл. 7.6, рис. 7.3, г) служат для запуска и прекращения выполнения программ,

Таблица 7.6

Мнемоника	Команда	Код
HALT	Останов	000000
WAIT	Ожидание прерывания	000001
RESET	Сброс внешних устройств	000005
CLN	Очистка N	000250
CLZ	Очистка Z	000244
CLV	Очистка V	000242
CLC	Очистка C	000241
CCC	Очистка всех разрядов (N, Z, V, C)	000257
SEN	Установка N	000270
SEZ	Установка Z	000264
SEV	Установка V	000262
SEC	Установка C	000261
SCC	Установка всех разрядов (N, Z, V, C)	000277
NOP	Нет операции	000240

а также синхронизации процессов обработки информации при построении многомашинных систем.

Современная методика программирования рекомендует при разработке программ использовать *принцип модульности*, согласно которому в достаточно сложной и большой программе выделяется ряд функционально

законченных групп операторов (модулей) и оформляется в виде подпрограмм.

В этом случае программа представляется в виде *головной программы* и *набора подпрограмм* (рис. 7.6). Головная программа осуществляет вызов подпрограм-

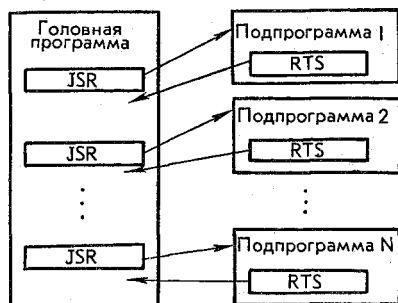


Рис. 7.6

мы, т. е. загружает в СК ее начальный адрес. Подпрограмма выполняет необходимые действия, и по их окончании осуществляется возврат в головную программу. В рассматриваемой системе команд DEC операции вызова подпрограммы и возврата из нее реализуются командами JSR и RTS.

Система команд Intel. Особенностью системы команд МП Intel 8080 (МП КР580ВМ80) является наличие нескольких адресных пространств. МП может обращаться к адресному пространству памяти объемом 64 К байт (16-разрядный адрес) и к адресным пространствам портов ввода объемом 256 байт (8-разрядный адрес) и портов вывода такого же объема. Это значит, что МП может обмениваться данными, например, с 5-й ячейкой памяти, с 5-м портом ввода и с 5-м портом вывода, т. е. с тремя разными физическими устройствами, имеющими один и тот же адрес — 5. Наличие отдельных адресных пространств памяти и ввода-вывода обычно означает, что из всего множества команд МП одни команды работают с памятью, другие — с вводом-выводом. Так, в системе команд МП КР580ВМ80, в частности, только в команде IN (ВВОД) МП обращается к адресному пространству ввода и только в команде OUT (ВЫВОД) — к адресному пространству вывода. Во всех остальных командах при внешних обращениях МП обращается к адресному пространству памяти.

В рассматриваемой системе команд используются следующие методы адресации: регистровый (операнд в памяти, адрес операнда — в команде); косвенный (операнд в памяти, адрес операнда — в регистровой паре); непосредственный (операнд указывается в команде); прямой; через указатель стека УС (операнд в стеке).

При *регистровой адресации* регистр-приемник и регистр-источник указываются 3-разрядным кодом:

0 0 0 — регистр В;
0 0 1 — регистр С;
0 1 0 — регистр D;
0 1 1 — регистр E;
1 0 0 — регистр H;
1 0 1 — регистр L;
1 1 1 — аккумулятор А.

Код 110 зарезервирован для обозначения команд, работающих с операндами, расположенными в памяти (М).

Если операнд 16-разрядный и расположен в регистровой паре (РП), то РП задается в команде с регистровой адресацией 2-разрядным кодом:

0 0 — пара В, С;
0 1 — пара D, E;
1 0 — пара H, L;
1 1 — регистр SP (УС) или пара А, F (PSW).

Регистровая пара А, F (F — регистр признаков) иногда обозначается в командах как PSW (от англ. program status word — слово состояния программы). На рис. 7.7 (обозначения аналогичны использованным на рис. 5.4) показана схема прямой адресации на примере команды пересылки из РОН С в аккумулятор А (MOV А, С). Выбор регистра осуществляется внутри МП, пересылка происходит по внутренней шине данных.

При *косвенной адресации* операнд расположен в памяти, но в команде явно не указывается его адрес. Адрес берется из регистровой пары H, L, называемой регистром косвенного адреса. На рис. 7.8 показана схема пересылки из регистра С в ячейку памяти при выполнении команды MOV М, С.

Если операнд указывается непосредственно в команде (во 2-м байте для 8-разрядных данных или во 2-м и 3-м — для 16-разрядных данных), то такая адресация называется *непосредственной*. Схема загрузки регистра С (MOV С, XX) приведена на рис. 7.9. После чтения

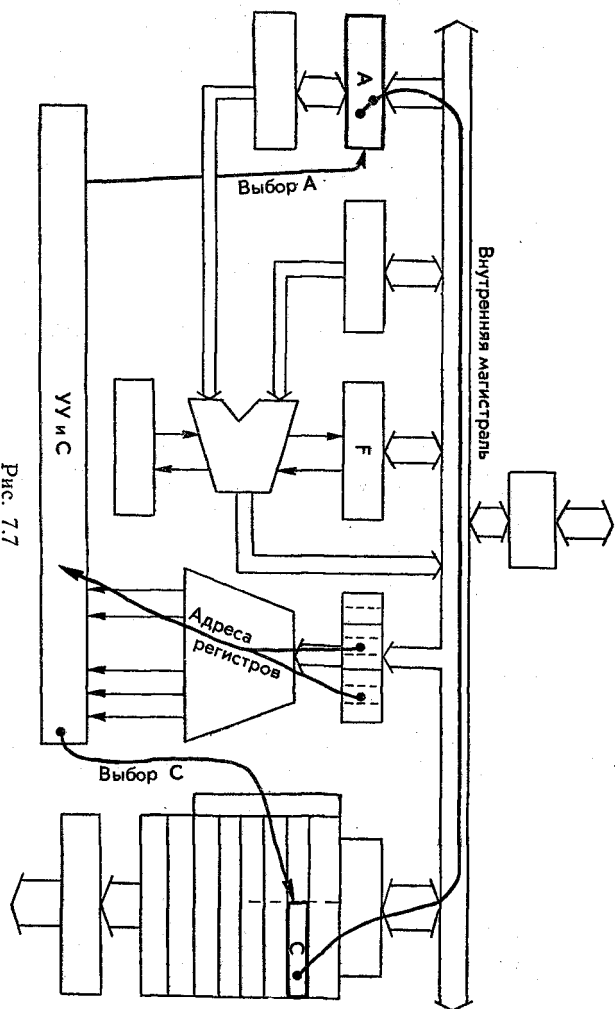


Рис. 7.7

и дешифрации 1-го байта команды, содержащего КОП, содержащее РС автоматически увеличивается на единицу, и теперь по нему определяется адрес 2-го байта, содержащего данные (XX_{16}).

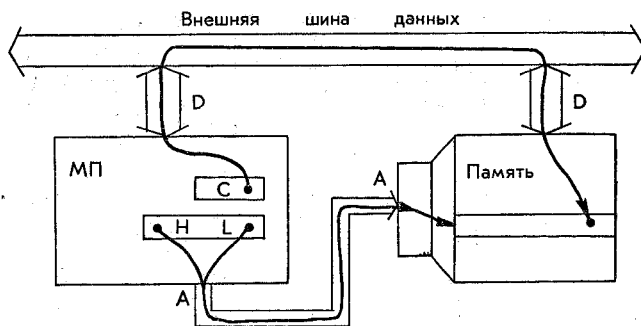


Рис. 7.8

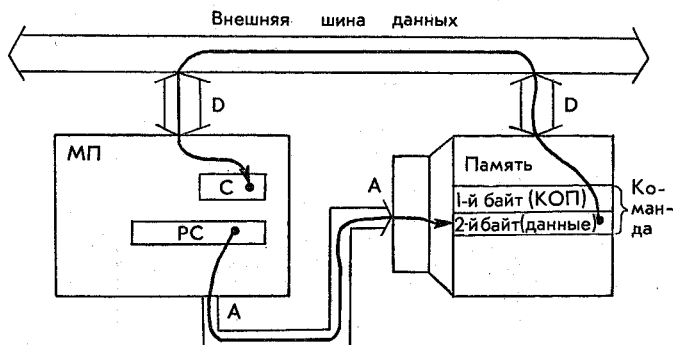


Рис. 7.9

Прямая адресация предполагает указание адреса ячейки памяти прямо в команде. Команда имеет 3-байтный формат, 2-й и 3-й байты содержат, соответственно, младшую и старшую половины 16-разрядного адреса. Пример прямой адресации приведен на рис. 7.10. Адрес из команды читается в МП (в регистр адреса РА), затем по этому адресу происходит обращение к ячейке. По команде STA <адрес> содержимое регистра А запоминается по указанному адресу.

Специальные команды используют область памяти, называемую *стеком*. Адрес очередной ячейки стека содер-

жится в указателе стека SP. При каждой записи в стек содержимое SP увеличивается на единицу, после каждого чтения из стека — уменьшается на единицу. Схема адресации через указатель стека приведена на рис. 7.11 (по

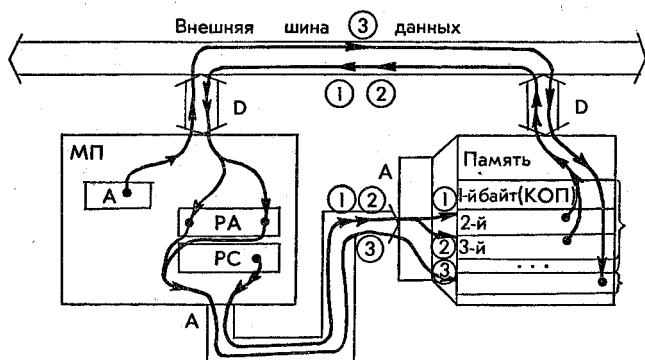


Рис. 7.10

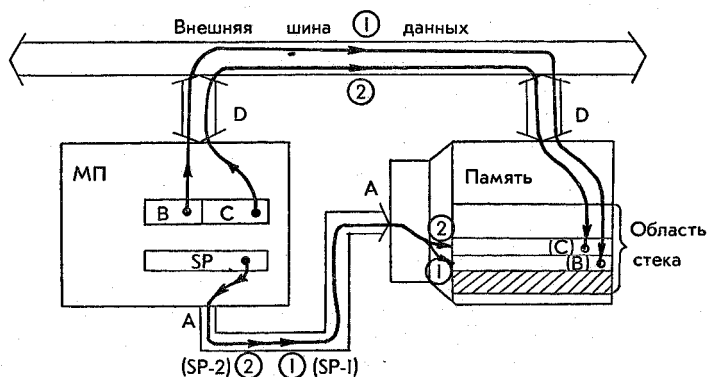


Рис. 7.11

команде PUSH В содержимое пары В, С запоминается в стеке).

Стек называют также *магазинной памятью*. Запись данных в стек можно сравнить со складыванием книг в стопку, одна на другую. Книга, которая была положена последней и оказалась сверху, будет взята первой. О таком стеке говорят, что он имеет дисциплину обслуживания LIFO (от англ. last input — first output — последним

вошел — первым вышел). На рис. 7.12, а — е показано изменение стека (и указателя стека) при обращении к нему по командам записи и чтения.

При рассмотрении команд МП КР580ВМ80 используют те же обозначения, что и в системе команд ДЕС.

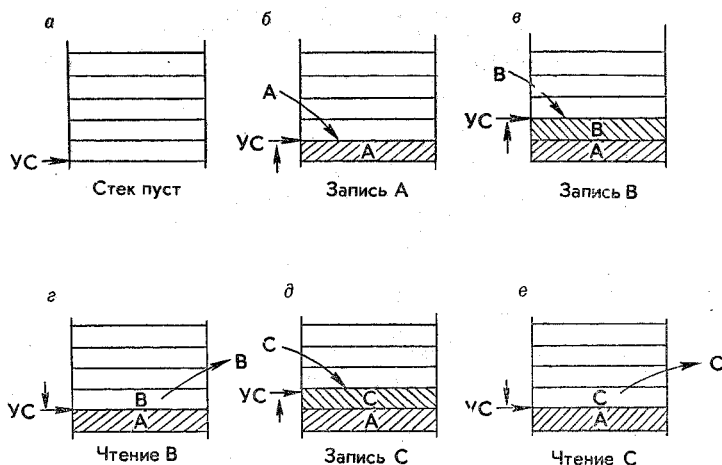


Рис. 7.12

Например: MOV A, B — пересылка из регистра B в аккумулятор A; ADD B — прибавить содержимое B к аккумулятору и результат занести в аккумулятор; MOV B, M — переслать содержимое ячейки памяти (адрес которой находится в регистровой паре H, L) в регистр B.

Команда пересылки имеет несколько модификаций:

MOV r ₁ , r ₂	(r ₁) ← (r ₂)
MOV r, M	(r) ← ((H, L))
MOV M, r	((H, L)) ← (r)

Запись ((H, L)) означает косвенную адресацию памяти через регистровую пару H, L. Непосредственную адресацию использует команда MVI, например, MVI C, XXH (переслать константу XX₁₆ в регистр C).

К арифметическим командам относят такие, как сложение (ADD) и вычитание (SUB). Источник одного из операндов и приемник результата — аккумулятор. Другой источник операнда указывается в команде, например ADD B ((A) ← (A) + (B)). В результате выполнения этих и других арифметических команд в РП записываются признаки результата.

Под логическими понимают команды, в которых над двоичными разрядами слова данных выполняются логические операции.

Логические команды в отличие от арифметических 8-разрядный операнд рассматривают не как единое слово, а как набор 8 разрозненных, самостоятельных двоичных разрядов. Логические команды выполняются поразрядно, например $ANA\ B\ ((A) \leftarrow (A) \wedge (B))$. Если $A = 11010110$, $B = 10000011$, то после $ANA\ B$ получится $A = 10000010$.

Группа команд переходов предназначена для реализации ветвящихся и циклических участков алгоритмов (рис. 7.13). В команде безусловного перехода JMP (от англ. *jump* — прыжок)

указывается адрес перехода (16 разрядов). Например, $JMP\ 4000H$ — переход к команде с адресом 4000_{16} .

Ветвлению соответствуют команды *условных переходов*. Например, «переход по нулю» $JZ\ 1000H$ — если $Z = 1$ (нулевой результат), то переход на 1000_{16} , если же $Z = 0$ (ненулевой результат), то выполняется следующая команда. Есть также команды JNZ — «переход по ненулевому результату» и другие, использующие признаки РП. К этой группе относятся также команды вызова подпрограмм и возврата из подпрограммы.

В системе команд МП КР580ВМ80 имеются команды работы со стеком (например, $PUSH\ B$ — записать в стек содержимое регистровой пары B, C), команды ввода-вывода (IN — ввод и OUT — вывод), команды управления (HLT (halt) — останов, NOP — пустая команда, EI и DI — разрешить прерывания и запретить прерывания).

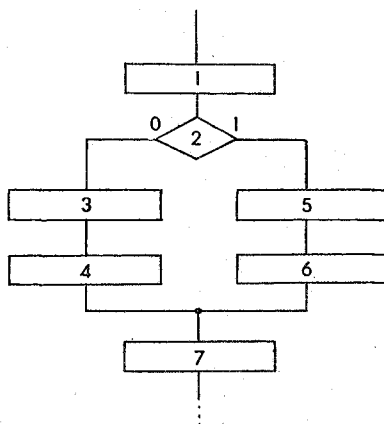


Рис. 7.13

7.3. Автоматизация программирования

Разработка программ в машинных кодах, в двоичной, восьмеричной или шестнадцатеричной системах счисле-

ния — трудоемкий процесс, требующий высокой квалификации программиста, досконального знания принципов построения аппаратной части микроЭВМ и особенностей выполнения каждой команды из системы команд. Программы в машинных кодах оптимальны по времени выполнения и затратам памяти. Но даже опытные программисты делают ошибки при их написании, а сам процесс программирования сравнительно долг. Трудность программирования в машинных кодах обусловлена непривычностью формы представления и обработки информации (двоичная, восьмеричная или шестнадцатеричная системы счисления).

Поэтому большое внимание уделяется автоматизации программирования. Основой автоматизации является использование языков программирования, которые представляют собой набор символов и систему правил образования и истолкования конструкций из этих символов, предназначенных для записи программ и данных:

Современные языки программирования близки к естественной форме восприятия информации человеком, просты в освоении и использовании. Однако написанная на таком языке программа не может быть непосредственно выполнена машиной. Ее необходимо транслировать, т. е. перевести в машинные команды, допустимые в системе команд данной микроЭВМ. Трансляция выполняется специальными программами-трансляторами.

Структура языка включает набор сгруппированных определенным образом операторов — совокупности символов, указывающих операцию и значение или местонахождение ее операндов. Особую группу среди операторов составляют служебные операторы, или *директивы*, используемые только для работы транслятора.

По способу трансляции и исполнения исходной программы трансляторы подразделяются на компиляторы и интерпретаторы.

Компилятор выполняет перевод текста исходной программы на язык машинных команд, или машинно-ориентированный язык. Полученная в результате компиляции программа называется *объектной*. Для трансляции широко используется набор типовых машинных процедур — программных блоков и подпрограмм, выполняющих определенные функции: ввод-вывод данных, преобразование данных во внутреннюю форму представления и обратно, арифметические операции, вычис-

ление математических функций и т. п. Объектная программа представляет собой совокупность таких программных блоков и ссылок на подпрограммы с использованием относительных адресов.

Окончательную сборку программы и «привязку» ее к конкретным адресам оперативной памяти осуществляет программа *редактор* (редактор связей). Результатом работы редактора является *загрузочная* (абсолютная) программа, готовая к загрузке в оперативную память микроЭВМ и выполнению.

Интерпретатор осуществляет последовательное преобразование в машинные команды и немедленное исполнение каждого оператора исходного языка.

В каких случаях эффективно использование компиляторов, а в каких — интерпретаторов? Пусть время исполнения интерпретируемой программы $T_{ин}$, а загрузочной (скомпилированной) — $T_{зг}$. Очевидно, что $T_{ин} > T_{зг}$, однако в случае компиляции следует учитывать разовые затраты времени на компиляцию и редактирование $T_{кр}$. Если программа исполняется $N_{исп}$ раз, то суммарное машинное время, требуемое программе, составляет $N_{исп} T_{ин}$ — в случае интерпретации и $N_{исп} T_{зг} + T_{кр}$ — в случае компиляции. Поэтому при $N_{исп} < T_{кр} / (T_{ин} - T_{зг})$ более выгодна интерпретация, а при $N_{исп} > T_{кр} / (T_{ин} - T_{зг})$ — компиляция. Это означает, что при многократном использовании программ более экономичным является процесс компиляции, а если программа исполняется редко и подвержена частым изменениям, то преимущество — на стороне интерпретации.

С целью сокращения времени разработки программного обеспечения для микроЭВМ используются и другие методы автоматизированной подготовки программ. Например, программное обеспечение, написанное для ЭВМ с системой команд S1, может быть автоматически «перенесено» на микроЭВМ с системой команд S2. Этот процесс называется эмуляцией, а программа, выполняющая его, — *эмулятором*.

Довольно часто программы для микроЭВМ транслируют на мини-ЭВМ с такой же системой команд, используя их более развитые программные средства. Такой процесс получил название кросс-трансляции.

7.4. Машинно-ориентированные языки

Общие сведения. Язык программирования, отражающий структуру ЭВМ, называется машинно-ориентированным. Среди машинно-ориентированных языков получили распространение АССЕМБЛЕРы и МАКРОАССЕМБЛЕРы, реализуемые одноименными трансляторами.

АССЕМБЛЕР включает в свой набор операторов систему команд микроЭВМ. Структура оператора АССЕМБЛЕРА максимально приближена к формату команды и содержит четыре поля:

МЕТКА: ОПЕРАЦИЯ ОПЕРАНДЫ; КОММЕНТАРИЙ

Поле МЕТКА предназначено для записи символического адреса (имени) оператора. Оно требуется, например, в случае, когда на оператор есть ссылка в программе. Обычно в качестве метки допускается применять начинающуюся с буквы алфавитно-цифровую последовательность длиной не свыше 8 символов.

Поле ОПЕРАЦИЯ содержит mnemonic обозначение кода операции машинной команды.

Поле ОПЕРАНДЫ отведено для одного или нескольких операндов. Если операндов несколько, то они разделяются запятыми. В качестве операндов могут быть числа, символические имена и выражения. В записи операндов могут использоваться специальные знаки (α , #, скобки и др.).

В поле КОММЕНТАРИЙ помещают текстовую информацию, не влияющую на выполнение программы и служащую для пояснения оператора.

Все перечисленные поля отделяются друг от друга по крайней мере одним пробелом. Поля МЕТКА и КОММЕНТАРИЙ часто отделяются знаками «:» и «;» соответственно.

	MOV R1,R2	; ПЕРЕСЫЛКА ДАННЫХ
	CLR R3	; ОЧИСТКА РЕГИСТРА
МЕТ:	TST @R4	; УСТАНОВКА ПРИЗНАКОВ
	MOV #1000,R6	; УСТАНОВКА ВЕРШИНЫ СТЕКА

МАКРОАССЕМБЛЕР является расширенным вариантом АССЕМБЛЕРА, дополнительно включающим такие средства, как макрокоманды. *Макрокоманда* — это оператор, который при трансляции заменяется последова-

тельностью других операторов языка. Эта последовательность называется *макрорасширением*. Макрорасширение не конструируется транслятором-макроассемблером заново, а берется из библиотеки стандартных макрорасширений языка.

Каждой макрокоманде соответствует *макроопределение* — совокупность операторов, используемых для формирования макрорасширения всякий раз, когда в исходном модуле встречается макрокоманда.

МАКРОАССЕМБЛЕР микроЭВМ «Электроника-60».

Алфавит языка содержит следующие символы: 1) прописные буквы латинского алфавита от А до Z (26 букв); 2) десятичные цифры от 0 до 9; 3) специальные знаки, знаки арифметических операций и разделители: @ % = # : ; , ' () « + — & ? пробел; 4) управляющие символы формата ВК (возврат каретки), ПС (перевод строки), ПФ (перевод формата) и ГТ (горизонтальная табуляция). В комментариях и некоторых директивах можно использовать прописные буквы русского алфавита.

Программа на языке МАКРОАССЕМБЛЕР состоит из последовательности строк (предложений), содержащих до 80 знаков (включая пробелы).

Синтаксическими элементами предложений являются символические имена и операции.

Символические имена (цифро-буквенные обозначения) могут быть трех видов: постоянные имена, имена пользователя и имена макрокоманд.

К *постоянным именам* относятся символические обозначения кодов операций в системе команд и директив МАКРОАССЕМБЛЕРА.

Именами пользователя являются имена, использованные программистом для обозначения меток и переменных. Имена, определенные пользователем, могут быть локальными и глобальными. *Локальные* имена используются только в том программном модуле, где они определены. *Глобальные* имена могут быть доступны всем программным модулям. Они описываются директивой .GLOBL во всех модулях, где используются и определяются.

Особое место в языке занимают имена регистров и СК. Регистр обозначается знаком %, а СК — точкой.

Z1	; РЕГИСТР 1
MOV Z1,Z4	; ПЕРЕСЫЛКА ИЗ РЕГИСТРА 1 В РЕГИСТР 4
MOV .,Z0	; В РЕГИСТР 0 ЗАПИСЫВАЕТСЯ КОД КОМАНДЫ MOV.,Z0
MOV #.,Z1	; В РЕГИСТР 1 ЗАПИСЫВАЕТСЯ АДРЕС КОМАНДЫ

Операции определяют действия над данными (константами и переменными). МАКРОАССЕМБЛЕР допускает использование констант трех типов: целые, вещественные и символьные.

Целые константы могут быть заданы в восьмеричной, десятичной и двоичной системах счисления. При этом перед числом в восьмеричной системе допускается записывать или не указывать букву О, в десятичной — букву D или после числа — точку, а в двоичной — букву В.

2541	;	ВОСЬМЕРИЧНОЕ ЧИСЛО
02541	;	ЭКВИВАЛЕНТНАЯ ЗАПИСЬ ВОСЬМЕРИЧНОГО ЧИСЛА
0199	;	ДЕСЯТИЧНОЕ ЧИСЛО
199.	;	ЭКВИВАЛЕНТНАЯ ЗАПИСЬ ДЕСЯТИЧНОГО ЧИСЛА
001100	;	ДВОИЧНОЕ ЧИСЛО
C151	;	ЗАПИСЬ ВОСЬМЕРИЧНОГО ЧИСЛА В ОБРАТНОМ КОДЕ.
	;	СООТВЕТСТВУЕТ КОДУ 177626

Вещественные константы задаются с помощью указателя F и преобразуются транслятором в двухбайтное число с плавающей запятой.

F1.0	;	ТОЧКА ОТДЕЛЯЕТ ЦЕЛУЮ ЧАСТЬ ОТ ДРОБНОЙ
F25.6		

Символьная константа — это один или два символа, закодированных в коде КОИ-7. Для обозначения одного символа используется апостроф, для двух — кавычки.

MOV # 'A, X0	;	КОД СИМВОЛА A
	;	(000101 В ВОСЬМЕРИЧНОЙ СИСТЕМЕ СЧИСЛЕНИЯ)
	;	ПЕРЕСЫЛАЕТСЯ В РЕГИСТР 0
MOV # "AB, X1	;	КОДЫ СИМВОЛОВ AB (041101) ПЕРЕСЫЛАЮТСЯ
	;	В РЕГИСТР 1

Предложения подразделяются на четыре вида: операторы машинных команд, директивы, операторы прямого присваивания и макрокоманды.

Операторы машинных команд транслируются в соответствующие им машинные коды, рассмотренные в § 7.2.

Директивы используются для управления процессом трансляции и выполнения различных функций, а именно: по управлению печатью листинга, распределению памяти, секционированию и объединению программ, резервированию памяти, записи данных в определенном формате. Директивы не порождают команд в машинном коде.

Директивы управления листингом используются для управления содержанием, форматом и формированием страниц листинга.

Директивы `.LIST` и `.NLIST` управляют печатью определенных символическими аргументами полей листинга. Если аргументы отсутствуют, то директивы `.LIST` и `.NLIST` подсчитываются транслятором и знак содержимого счетчика служит указателем вывода листинга (первоначально в счетчике записан 0; каждая директива `.LIST` добавляет, а `.NLIST` вычитает единицу): при отрицательном значении строки программы не распечатываются (за исключением предложений, содержащих ошибки), положительное значение вызывает распечатку строк.

<code>.LIST</code>	; ПЕЧАТАТЬ СЛЕДУЮЩУЮ СТРОКУ
<code>CLR @R0</code>	
<code>.NLIST</code>	; ПЕЧАТАТЬ СЛЕДУЮЩУЮ СТРОКУ. СЧЕТЧИК РАВЕН 0
<code>MOV R1,R2</code>	
<code>.NLIST</code>	; НЕ ПЕЧАТАТЬ СЛЕДУЮЩУЮ СТРОКУ
<code>ASR R2</code>	

Директивы `.LIST` и `.NLIST` предназначены в основном для выборочной распечатки макрорасширений.

Наличие аргументов в этих директивах не изменяет значение счетчика печати. В качестве аргументов могут быть использованы только строго определенные имена, имеющие соответствующее смысловое содержание.

Следующие имена, будучи аргументами, обеспечивают управление распечаткой: `BEX` — второго и третьего слов двоичного кода; `CND` — блоков условной трансляции в случае, если не выполнены условия их трансляции; `COM` — комментария; `LD` — всех директив управления листингом, в которых нет аргументов; `LOC` — счетчика адресов; `MC` — вызовов макрокоманд; `MD` — макроопределений; `ME` — макрорасширений; `MEB` — двоичного кода макрорасширения; `SEQ` — порядковых номеров строк; `SRC` — исходного текста; `SYM` — таблицы символов программы; `TOC` — оглавления при первом проходе трансляции; `WIN` — создаваемого двоичного кода. Имя `TTM` обеспечивает управление форматом выходных строк листинга.

Директива `.TITLE` используется для назначения имени объектному модулю.

```
.TITLE PROG1
Объектному модулю присваивается имя PROG1
```

Если директива `.TITLE` отсутствует, то объектному модулю присваивается имя `MAIN`.

Директива .SBTTL предназначена для формирования оглавления листинга и обозначения страниц.

.SBTTL CONDITIONAL ASSEMBLIES

Текст, указанный после имени директивы, будет печататься в каждой странице листинга

Директива .PAGE вызывает печать следующего за ней текста с новой страницы листинга.

Директива .REM позволяет ввести комментарий в исходную программу без использования символа «;».

.REM ПОДПРОГРАММА 1

Директивы режима трансляции предназначены для управления трансляцией и печатью таблицы перекрестных ссылок. К ним относятся директивы:

.ENABL — разрешить выполнение определенной функции;

.DSABL — запретить выполнение указанной функции;

.CROSS и .NOCROSS — соответственно разрешают и запрещают указанные действия по печати таблицы перекрестных ссылок.

Директивы задания данных предназначены для записи данных в различных видах:

.BYTE — в двоичном виде;

.WORD — в виде последовательности слов, расположенных в соседних ячейках памяти;

.ASCII и .ASCIZ — в виде последовательности кодов КОИ-7;

.RAD50 — в виде кода RADIX-50, в котором каждые три символа упаковываются в одно слово;

.PACKED — используется для упаковки десятичных данных в два байта;

.FLT2 и .FLT4 — для записи десятичных чисел в форме с плавающей запятой одинарной (четыре байта) и двойной (восемь байтов) точности.

Пусть, например, счетчик адреса команд равен 1000 в восьмеричной системе счисления, тогда

.BYTE 23	; ЗАПИСЫВАЕТ ВОСЬМЕРИЧНОЕ ЧИСЛО 23 В
	; ЯЧЕЙКУ 1000
.WORD 25,4,CAL	; 25, 4 И ЗНАЧЕНИЕ ПЕРЕМЕННОЙ CAL ЗАПИСЫ-
	; ВАЮТСЯ В ЯЧЕЙКАХ 1000, 1002 И 1004
ASCII /СТРОКА СИМВОЛОВ/	; ЗАПОМИНАЕТ ДВОИЧНОЕ ПРЕДСТАВЛЕНИЕ
	; ЗАПИСИ "СТРОКА СИМВОЛОВ" В ПОСЛЕДОВА-
	; Тельных байтах
.FLT2 ARG1,ARG2	; ПЕРЕМЕННЫЕ ARG1 И ARG2 ПРЕДСТАВЛЯЮТСЯ
	; В ФОРМЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

Директивы управления счетчиком адреса:

.EVEN — добавляет единицу к текущему значению СК в случае его нечетности. Напомним, что команда в системе DEC всегда имеет четный адрес. Поэтому директива **.EVEN** применяется обычно после использования директив, приводящих к нечетному значению СК (**.ASCII**, **.ASCIZ**, **.BYTE**):

```
.ASCII /СИСТЕМА/  
.EVEN
```

.ODD — добавляет единицу СК в случае его четности;

.BLKB и **.BLKW** — резервируют области памяти в объектном модуле в байтах и словах соответственно;

.LIMIT — резервирует 2 слова памяти. Первое слово предназначено для записи младшего адреса загрузочного модуля, второе — для записи адреса первого свободного слова, следующего за загрузочным модулем (старший адрес загрузочного модуля + 2).

```
.BLKB 5 ; РЕЗЕРВИРОВАНИЕ 5 БАЙТ  
.BLKW 7 ; РЕЗЕРВИРОВАНИЕ 7 СЛОВ  
.LIMIT
```

Директивы секционирования дают возможность управлять распределением памяти для программы во время ее связывания. Все признаки, введенные с помощью этих директив, впоследствии передаются редактору связей. К директивам секционирования относятся:

.PSECT — создание программной секции (программного блока);

.ASECT — создание абсолютной программной секции;

.CSECT — создание относительной (переменной) программной секции;

.SAVE и **.RESTORE** — используются для записи в стек и выборки из стека аргументов директивы **.PSECT**.

Директива конца **.END** предназначена для указания логического конца исходного модуля.

Директивы условной трансляции **.IF** и **.ENDC**, **.IFF**, **.IFT**, **.IFTF**, **.IIF** применяют для получения различных вариантов объектной программы из одной исходной.

Оператор прямого присваивания имеет формат

МЕТКА:ИМЯ=ВЫРАЖЕНИЕ;КОММЕНТАРИЙ

Метка и комментарий не являются обязательными. При трансляции вычисляется значение арифметико-логическо-

го выражения, которое присваивается имени переменной. Имя и его значение заносятся в специальную символьную таблицу.

Выражение — это один или несколько термов, соединенных знаками двухместных операций. Термом может быть: константа; символическое имя; выражение, заключенное в угловые скобки, имеющие тот же смысл, что и круглые в записи математических формул; символическое имя или константа, которой предшествует знак одноместной операции.

RO=20	; ИМЕНИ RO ПРИСВОИТЬ ЗНАЧЕНИЕ РЕГИСТРА 0
SP=X6	; В КАЧЕСТВЕ УКАЗАТЕЛЯ СТЕКА ВЫБРАТЬ РЕГИСТР 6
.=.+400	; УВЕЛИЧИТЬ СК НА 400
A=129.	; ПЕРЕМЕННАЯ A ПРИСВАИВАЕТСЯ ЗНАЧЕНИЕ 129
	; В ДЕСЯТИЧНОЙ СИСТЕМЕ СЧИСЛЕНИЯ

Макроккоманды, определенные пользователем, должны быть описаны директивой `.MACRO`, имеющей формат

МЕТКА: `.MACRO` ИМЯ, СПИСОК АРГУМЕНТОВ

и обозначающей начало макроопределения макрокоманды «ИМЯ». Конец макроопределения обозначается директивой `ENDM ИМЯ` (здесь ИМЯ — необязательный аргумент).

Например, часто используемый программный блок сохранения регистров в стеке может иметь макроопределение:

```
.MACRO SAVE
MOV R0,-(SP)
MOV R1,-(SP)
MOV R2,-(SP)
MOV R3,-(SP)
.ENDM
```

В тексте объектного модуля основной программы

```
CLR R0
MOV R0,R1
ADD R3,R0
SAVE
CLR R0
```

макрокоманда `SAVE` будет заменена макрорасширением, сформированным из макроопределения.

Пример 7.1. Обнуление ячеек памяти с 0 по 77776.

```

        .TITLE RESET
        R0=X0
        R1=X1
        CLR R0                ; УСТАНОВИТЬ НАЧАЛЬНОЕ ЗНАЧЕНИЕ
        MOV $100000,R1       ; УСТАНОВИТЬ КОНЕЧНОЕ ЗНАЧЕНИЕ
M1:     CLR (R0)+             ; ОБНУЛЕНИЕ ЯЧЕЙКИ
        CMP R1,R0            ; R1-R0
        BPL M1               ; ЕСЛИ R1>R0, ИДТИ К M1
        .END

```

Пример 7.2. Программа ввода с клавиатуры и печати знака на экране дисплея — пультавого терминала.

```

        .TITLE INPUT
        R0=X0
        R1=X1
        R2=X2
        RSKL=177560          ; RSKL - РЕГИСТР СОСТОЯНИЯ КЛАВИАТУРЫ
        RDKL=177562          ; RDKL - РЕГИСТР ДАННЫХ КЛАВИАТУРЫ
        RSD=177564           ; RSD - РЕГИСТР СОСТОЯНИЯ ДИСПЛЕЯ
        RDD=177566           ; RDD - РЕГИСТР ДАННЫХ ДИСПЛЕЯ
        .MACRO OPPOS ARG;    ; ОПРОС ГОТОВНОСТИ
M1:     TSTB @ARG             ; УСТАНОВКА ПРИЗНАКОВ ПО МЛАДШЕМУ БАЙТУ
        BGE M1               ; ЕСЛИ УСТРОЙСТВО НЕ ГОТОВО (РАЗРЯД В НЕ
        .ENDM                ; УСТАНОВЛЕН), ИДТИ К M1
        OPPOS RSKL
        MOV @RDKL,R0         ; ВВОД КОДА СИМВОЛА R0
        OPPOS RSD
        MOV R0,@RDD          ; ВЫВОД КОДА СИМВОЛА НА ЭКРАН
        .END

```

Пример 7.3. Подпрограмма преобразования числа с фиксированной запятой в число с плавающей запятой. Диапазон числа X: $-1 < X < 1$. Исходное и полученное числа передаются через стек. Для старшей части числа в стеке резервируются ячейки памяти. При обращении к подпрограмме в стек заносится адрес следующей команды вызывающей программы.

```

        .TITLE FIXFL
        .GLOBAL FIXFL
        .CSECT
        R0=X0
        R1=X1
        R5=X5
        SP=X6

        FIXFL:MOV @SP,R0      ; СОХРАНЕНИЕ АДРЕСА КОМАНДЫ
                                ; ВЫЗЫВАЮЩЕЙ ПРОГРАММЫ
                                ; АНАЛИЗ ЗНАКА И НУЛЯ
        TST 2(SP)
        BEQ F4                ; F4, ЕСЛИ ЧИСЛО = 0
        BPL F1                ; F1, ЕСЛИ ЧИСЛО > 0
        NEG 2(SP)
        MOV $146000,R1        ; ПРЕДСТАВЛЕНИЕ ЗАДАННОГО ЧИСЛА В СТЕКЕ
        BR M1                 ; С ПЗ С ИЗБЫТКОМ 2
F1:     MOV $4600,R1           ; В СТЕПЕНИ -9
M1:     MOV R1,@SP            ; ЗАПИСЬ В СТЕК СТАРШЕЙ ЧАСТИ
                                ; ЧИСЛА С ИЗБЫТКОМ

```

CLR -(SP)	; ЗАПИСЬ ЧИСЛА В СТЕК
MOV R1, -(SP)	; ИЗБЫТКА 2 В СТЕПЕНИ -9
FSUB SP	; ВЫЧИТАНИЕ ИЗБЫТКА ИЗ ЧИСЛА
	; ПРИ ВЫПОЛНЕНИИ ОПЕРАЦИИ
	; SP УВЕЛИЧИВАЕТСЯ НА 4 И
	; ПРОИСХОДИТ НОРМАЛИЗАЦИЯ РЕЗУЛЬТАТА
BR RET	
F4: CLR ESP	; ЗАПИСЬ МИНИМАЛЬНОГО
CLR 2(SP)	; ЧИСЛА 2 В СТЕПЕНИ -128
RET: MOV R0, -(SP)	; ВОССТАНОВЛЕНИЕ АДРЕСА КОМАНД
RTS R5	; ВЫЗЫВАЮЩЕЙ ПРОГРАММЫ
.END	

7.5. Машинно-независимые языки

Машинно-независимые языки не ориентированы на конкретную микроЭВМ и для описания процесса вычислений они обладают более привычной для человека символикой, чем машинно-ориентированные языки.

Операторы таких языков обозначают, как правило, крупные вычислительные процедуры и транслируются не в одну, а в серию машинных команд. Поэтому часто машинно-независимые языки называют языками высокого уровня. Огромным преимуществом этих языков является то, что написанные на них программы могут исполняться на различных ЭВМ с разными системами команд, если ЭВМ снабжены соответствующими трансляторами. Кроме того, при использовании языков высокого уровня повышается производительность труда программиста, уменьшается число ошибок при разработке программ.

Машинно-независимые языки в основном используются при создании прикладных программ, для которых затраты машинного времени и оперативной памяти не являются определяющими. Это связано с тем, что объектные программы, полученные в результате трансляции с языка высокого уровня, характеризуются большим числом машинных команд и, следовательно, временем исполнения программы, чем написанные человеком на АССЕМБЛЕРЕ. Причина заключается в том, что для генерации машинных команд при трансляции операторов исходной программы используется универсальный набор блоков машинных команд. В силу своей универсальности этот набор для каждой конкретной исходной программы является избыточным.

Среди машинно-независимых языков различают процедурно- и проблемно-ориентированные языки.

К *процедурно-ориентированным* языкам относят алгоритмически универсальные языки, пригодные для программирования любых задач. В то же время каждый

конкретный процедурно-ориентированный язык имеет средства проблемной специализации (достаточно широкой). В микроЭВМ получили распространение БЕЙСИК, ФОКАЛ, СИ, ПАСКАЛЬ, ПЛ/М, АДА.

Проблемно-ориентированные языки предназначены для описания задач узкой предметной области. Примерами могут служить языки моделирования объектов СТАМ, МОДИС, ФОРΟΣ и др.

Рассмотрим один из процедурно-ориентированных языков.

Язык БЕЙСИК (от англ. beginner's all purpose symbolic instruction code — BASIC) был задуман разработчиками как язык, доступный для усвоения студентами гуманитарных вузов. Однако простота грамматики и синтаксиса, сходство с ФОРТРАНОм привели к широкому использованию языка для математических и научно-технических расчетов.

В настоящее время имеется ряд модификаций (версий) БЕЙСИКа, различающихся не только системой команд, для которой написан транслятор, но и возможностями самого языка. Простые версии имеют 30—40 операторов, более сложные — несколько сотен.

Для языка БЕЙСИК разработаны трансляторы компилирующего и интерпретирующего типов. При работе с компилятором исходная программа проходит на ЭВМ три рассмотренных ранее этапа: компиляцию, редактирование и выполнение. Интерпретатор допускает работу в двух режимах: программном и непосредственного счета.

В программном режиме каждая строка программы нумеруется целым положительным десятичным числом. Строка может содержать один или несколько операторов, разделенных специальным знаком (двоеточие «:», наклонная черта «\» и др. в зависимости от модификации транслятора). Последовательность пронумерованных строк составляет программу.

В режиме непосредственного счета строка вводится в микроЭВМ без номера и немедленно исполняется. Этот режим используется для простых вычислений и при отладке программ.

Алфавит языка БЕЙСИК содержит 26 прописных латинских букв, цифры от 0 до 9, точку, запятую, кавычки, знаки арифметических операций (сложение «+», вычитание «—», умножение «*», деление «/», возведение в степень «^»), отношения двух величин (больше «>»,

меньше « $<$ », равно « $=$ »), скобки и другие специальные знаки.

Данные в языке представляются в виде констант и переменных.

Константы — это постоянные величины, которые при выполнении программы не изменяются. В языке БЕЙСИК различают: числовые константы и константы типа «строка символов» — литералы.

Обычно числовые константы, содержащие менее 8 цифр или имеющие десятичную экспоненту с буквой «Е», являются константами одинарной точности. Их внутреннее представление выглядит следующим образом: целые числа N , лежащие в диапазоне $-32768 \leq N \leq 32767$, изображаются в виде двухбайтных двоичных чисел с фиксированной запятой. Числовые константы одинарной точности, лежащие в диапазоне от $(10E-38)$ до $(10E+38)$ по модулю, представляются в виде четырехбайтных двоичных чисел с плавающей запятой.

Числовые константы, содержащие более 7 цифр или имеющие десятичную экспоненту с буквой «D», являются числовыми константами двойной точности. Они представляются в виде 8-байтного числа с плавающей запятой.

Иногда для выделения констант одинарной точности используют восклицательный знак «!», а двойной — знак «#».

Для записи числовых констант может быть использована шестнадцатеричная система счисления (числа представляются в форме $\&HN...N$) или восьмеричная (числа представляются в форме $\&ON...N$ или $\&N...N$, где N — шестнадцатеричные или восьмеричные цифры). Шестнадцатеричные константы могут лежать в диапазоне от H0000 до HFFFF. Во внутреннем изображении они представляют собой целые числа и имеют следующие десятичные эквиваленты:

$\&H0000... \&H7FFF$
 $\&H8000... \&HFFFF$

$0...+32767$
 $-32768...-1$

Восьмеричные числа лежат в диапазоне от 0 до 177777, воспринимаются и обрабатываются как целые числа, имеют следующие десятичные эквиваленты:

$\&000000... \&077777$
 $\&100000... \&177777$

$0...+32767$
 $-32768...-1$

Константы типа «строка символов» представляют собой произвольную последовательность допустимых в языке символов, заключенную в кавычки или апострофы. Максимальное число символов в разных версиях может быть различным и достигает 255.

Числовые константы одинарной точности

44.3 20.51! -1.078E07

Числовые константы двойной точности

1234567890 12345.678901 20.5# -1.078D-07

Шестнадцатеричные константы

&H2A70 &HCAB1

Восьмеричные константы

&O12345 &I2345

Константы типа "строка символов"

"123.45 MARK" 'БЕЙСИК'.

Переменные — это данные, значения которых в программе могут изменяться. Они идентифицируются именами. В языке БЕЙСИК различают переменные типа «строка символов», числовые переменные, указатели функций.

Имена переменных всегда начинаются с буквы, за которой могут следовать буквы или цифры. В зависимости от версии языка максимальное число знаков имени может составлять от 2 до 40. Имена переменных заканчиваются одним из следующих знаков, обозначающих тип переменной: α — обозначение переменной типа «строка символов», например A10 α ; % — обозначение INTEGER-переменной (целое двухбайтное число), например A1%; ! — обозначение REAL-переменной (действительное число) одинарной точности (4 байта), например R17! («!» может быть опущен); # — обозначение REAL-переменной двойной точности (8 байт), например ALPHA#.

Имена переменных, которые начинаются с FN, обозначают функции. Числовые переменные одинарной точности и переменные типа «строка символов» определяются контекстно, т. е. объявляются явно с их первым использованием. Если после имени переменной идентификатор типа не установлен, то переменная полагается действительного типа одинарной точности.

Правильные имена:

AX 15Z DIANE BEZ BEZ! ENDWERT

Неправильные имена:

1AX - обозначение переменной всегда начинается с буквы

AX1 - обозначение типа переменной всегда стоит в конце

FN# - последовательность знаков FN является зарезервированной, в качестве обозначения переменной использоваться не может

END - недопустимое обозначение, т.к. END в языке БЕЙСИК является зарезервированным словом.

Интерпретатор с языка БЕЙСИК включает следующие элементы: директивы, операторы, встроенные функции.

Директивы (служебные операторы) устанавливают определенные режимы работы интерпретатора, управляют ходом вычислительного процесса, осуществляют редактирование и отладку программ. Директивы выполняются сразу же после ввода их с пульта. Отдельные директивы могут выполняться и по программе. Список директив БЕЙСИК-интерпретатора для ЭВМ «ROBOTRON 1715» (прил. 1) иллюстрирует возможности интерпретатора, характерные для многих версий языка, хотя для других ЭВМ имена директив могут быть другими. В нижеприведенных примерах использования директив <ET> означает нажатие клавиши конца ввода строки программы (ET). Иногда эта клавиша называется CR/LF, BK и т. п.

AUTO 20,10 <ET>

Обозначает автоматическую генерацию номеров строк программы после каждого нажатия клавиши <ET>. Нумерация начинается с номера 20 с шагом 10. Если параметры не заданы, то полагается AUTO 10,10.

SYSTEM <ET>

Выполняется выход в операционную систему. При этом вся информация, размещенная в ОЗУ, стирается.

LIST 10-300 <ET>

(LLIST 10-300 <ET>)

На дисплей (печатающее устройство) выводится текст программы начиная с 10 и кончая 300 строкой. Для вывода программы или ее фрагментов может также использоваться следующая символика:

LIST +200 <ET> Вывод строк, начиная с номера 200

LIST -200 <ET> Вывод всех строк до номера 200

включительно

LIST 200 <ET> Вывод строки с номером 200

LIST <ET> Вывод всей программы.

DELETE 20-150 <ET> Стираются строки программы с номерами 20-150

DELETE 40 <ET> Стирается строка программы с номером 40

DELETE -180 <ET> Стираются строки программы до номера 180 включительно.

RENUM 500,100,10 <ET>

Строки программы, начиная с номера 100, нумеруются числами, начинающимися с 500 с шагом 10.

RENUM <ET> Строки программы нумеруются, начиная с 10 с шагом 10.

SAVE "PROGRAM" <ET>

Программа под именем PROGRAM сохраняется во внешнем запоминающем устройстве. После выполнения этой директивы в ОЗУ микроЭВМ можно загружать новую программу, например PR1, сохраненную ранее такой же директивой:

LOAD "PR1" <ET>.

RUN <ET>

Загрузка и запуск программы. Допускаются также модификации директив, обеспечивающие запуск программы с именем PR1:

RUN "PR1" <ET>

или с заданного номера строки, например 500:

RUN 500 <ET>.

Операторы задают микроЭВМ необходимые операции по обработке данных. Полный список операторов версии языка БЕЙСИК для «ROBOTRON 1715» приведен в прил. 2. Рассмотрим операторы, которые составляют основу языка и встречаются практически во всех его версиях. (В квадратные скобки заключены необязательные элементы конструкции оператора.)

Оператор присваивания имеет вид

N [LET] ИМЯ=ВЫРАЖЕНИЕ <ET>

где N — номер строки, ИМЯ — имя переменной, ВЫРАЖЕНИЕ — переменные, константы, встроенные функции или переменные и константы и встроенные функции, объединенные знаками арифметических операций и (или) круглыми скобками.

Например, оператором

10 LET A=2

переменной A присваивается значение 2. Обозначение LET в некоторых версиях языка допускается не указывать:

10 A=2

Оператором

10 A=X*Y+2/(Z-SQR(W+V))

переменной A присваивается значение арифметического выражения

$$A = xy + \frac{2}{z - \sqrt{w + v}}.$$

Скобки задают порядок выполнения операций такой, какой он принят при записи обычных математических формул.

Машинно-независимые языки широко оперируют такими структурами данных, как массивы и наборы. Под *массивом* понимается совокупность однотипных данных, объединенных под одним именем. Понятие «массив» означает по существу то же, что и «матрица» в математике. Современные версии языков допускают использование одномерных, двумерных, трехмерных массивов и массивов большей размерности (до 255). Каждый элемент массива имеет свой номер (или номерá — в случае многомерных массивов), указываемый в скобках. Например, элементы матриц A_2 , $ALFA_{3,7}$, AS_{ij} обозначают $A(2)$, $ALFA(3,7)$, $AS(I, J)$ соответственно.

В отличие от массива *набор данных* предполагает совокупность записей и констант, выступающую как единое целое при хранении и поиске данных. В общем случае данные, объединенные в набор, могут быть разнотипными.

Оператор описания массивов DIM (от англ. dimension — размерность):

N DIM ИМЯ (ВГП) [, ИМЯ (ВГП)]...

где ИМЯ — имя массива, образуемое по тем же правилам, что и имена переменных; ВГП — верхние границы параметров массива (максимальное число элементов массива по каждому индексу); многоточие означает возможность многократного повторения. Оператор DIM — невыполняемый, т. е. не порождает машинных команд, а дает лишь информацию транслятору о том, какие массивы используются в программе. Транслятор резервирует для массивов необходимое число ячеек памяти.

10 DIM A(20), B(40,60)

Резервирование места в памяти для одномерного массива A, состоящего из 20 элементов, двумерного массива B, состоящего из 40×60 элементов (40 — число строк, 60 — число столбцов).

Оператором DATA данные организуются в наборы:

N DATA K [, K]...

где K — константа одного из типов, допустимых в языке БЕЙСИК.

20 DATA 10,20,30,40,"ЦЕЛЫЕ ЧИСЛА"

Целые числа 10, 20, 30 и 40 объединены в один набор с литералом «ЦЕЛЫЕ ЧИСЛА».

Оператор ввода данных INPUT

Н INPUT [ЛИТЕРАЛ] ИМЯ [,ИМЯ]...

где ИМЯ — имя переменной или элемента массива. Оператор позволяет вводить данные с клавиатуры и присваивает введенные значения переменным или элементам массива с указанными именами. Если после названия оператора задан литерал, то перед вводом значений переменных содержание литерала выдается на экран дисплея. При отсутствии литерала на экран выводится вопросительный знак.

```
10 INPUT A
```

Ввод с клавиатуры значения переменной А после появления на экране дисплея знака «?». Для более содержательных пояснений при вводе используется формат

```
10 INPUT "A=",A
```

Значение переменной А вводится после появления надписи «А=».

```
10 INPUT "ВВЕДИТЕ ЧИСЛА А,В,С",А,В,С
```

Числа А, В, С последовательно вводятся после появления надписи «ВВЕДИТЕ ЧИСЛА А, В, С». Окончание ввода каждого числа индицируется нажатием (ЕТ).

Оператор вывода данных PRINT

Н PRINT ИМЯ [, ИМЯ]...

позволяет выводить значения переменных или элементов массива с именами ИМЯ на экран дисплея. В некоторых версиях языка вместо имени может быть арифметическое выражение. Каждое значение выводится со смещением на 14 знаковых позиций вправо. Если элементы списка разделены знаком «;» или пробелом, происходит вывод без смещения. Каждый оператор PRINT выполняет вывод с новой строки.

```
10 PRINT A,В,С
```

На экран дисплея последовательно выводятся значения переменных А, В, С.

```
10 A=2:В=4:С=5
20 PRINT A,В,С
RUN
2           4           5
```

Вывод текстовых констант и переменных осуществляется аналогично:

```
10 AX="ПРОГРАММА"  
20 PRINT "ЗНАЧЕНИЕ ПЕРЕМЕННОЙ AX=";AX  
RUN  
ЗНАЧЕНИЕ ПЕРЕМЕННОЙ AX=ПРОГРАММА
```

Оператор чтения данных из набора READ

Н READ ИМЯ [, ИМЯ]...

последовательно извлекает элементы данных из набора, описанного оператором DATA, и присваивает их значения переменным или элементам массива, имена которых перечислены в операторе.

```
10 READ A,B,C  
20 E=A+B+C  
30 DATA 2.1,3.2,4.3  
40 PRINT A,B,C,E  
RUN  
2.1          3.2          4.3          9.6
```

Данные 2,1; 3,2; 4,3; 5,2 организованы в набор оператором DATA. Оператор READ последовательно присваивает переменным A, B, C, D значения, описанные в операторе DATA. Оператор DATA — невыполняемый и может стоять в любом месте программы. В программе может быть несколько операторов DATA. В этом случае вначале считываются данные из набора DATA с меньшим номером, а затем — по мере возрастания номеров.

Оператор RESTORE осуществляет возврат к первому значению набора данных, соответствующего меньшему номеру оператора DATA:

```
10 READ A,B  
20 RESTORE  
30 READ C,D  
40 DATA 2,3  
50 DATA 9,10  
60 PRINT A,B,C,D  
RUN  
2          3          2          3
```

Оператор *условного перехода* в общем виде может быть записан следующим образом:

IF <A>R THEN N1 ELSE N2

или

IF <A>R GOTO N1 ELSE N2

Если справедливо соотношение R для выражений <A>

и $\langle B \rangle$, то выполняется переход к строке программы с номером N1, в противном случае — к N2. Если слово ELSE отсутствует, то в случае невыполнения соотношения R переход осуществляется к следующей строке. В качестве соотношений R принимаются следующие: = (равно), \neq (не равно), < (меньше), > (больше), \leq (меньше или равно), \geq (больше или равно).

```
10 IF A>B THEN 200 ELSE 400
20 IF A+B<=C+D GOTO 100
30 GOTO 500
```

Здесь же изображен оператор безусловного перехода GOTO, осуществляющий переход к строке 500.

Оператор цикла с фиксированным числом проходов имеет следующую конструкцию:

```
FOR ИМЯ=НАЧ TO КОН STEP ШАГ
ТЕЛО ЦИКЛА
NEXT ИМЯ
```

Здесь ИМЯ — имя переменной, изменяющейся от начального значения НАЧ до конечного КОН с шагом ШАГ. Если слово STEP в операторе FOR опущено, предполагается, что шаг равен единице. Оператор NEXT осуществляет увеличение переменной ИМЯ на ШАГ и возврат к началу тела цикла.

```
10 FOR I=1 TO 10
20 PRINT I;" ";
30 NEXT I
RUN
1 2 3 4 5 6 7 8 9 10
```

```
10 DIM A(10)
20 FOR I=1 TO 10 STEP 2
30 READ A(I)
40 NEXT I
50 DATA 10,20,30,40,50
60 FOR I=1 TO 10
70 PRINT A(I);" ";
80 NEXT I
RUN
10 0 20 0 30 0 40 0 50 0
```

По директиве RUN элементам массива A(I) присваивается значение 0. Затем при помощи цикла 20—40 нечетным элементам присвоены значения 10, 20, 30, 40, 50 соответственно. Во втором цикле 60—80 десять элементов массива A(I) выводятся на печать.

Встроенные функции — элементы языка, расширяющие его возможности. Они дополняют набор опе-

раторов и по существу определяют проблемную ориентацию интерпретатора. Перечень встроенных функций версии языка БЕЙСИК для микроЭВМ «ROBOTRON 1715» представлен в прил. 3.

Пример 7.4.

```
10 A=3.14/2
20 B=SIN(A)
30 PRINT "B="B
RUN
B=1
```

Вычисляется значение функции $\sin \pi/2$.

Пример 7.5.

```
10 X=3:Z=4
20 Y=SQR(X^2+Z^2)
30 PRINT "Y="Y
RUN
Y=5
```

Вычисляется значение квадратного корня из выражения $x^2 + z^2$.

Пример 7.6. Программа сортировки элементов массива КК(I) в порядке возрастания. Результирующий (ранжированный) массив имеет имя КК1.

```
10 REM СОРТИРОВКА ЭЛЕМЕНТОВ МАССИВА
20 DIM KK(10),KK1(10)
30 FOR I=1 TO 10
40 READ KK(I)
50 NEXT I
60 REM ПОИСК МАКСИМУМА
70 MAX=KK(1)
80 FOR I=1 TO 10
90 IF KK(I)>=MAX THEN MAX=KK(I)
100 NEXT I
110 REM ПОИСК МИНИМУМА
115 FOR J=1 TO 10
120 MIN=MAX
130 FOR I=1 TO 10
140 IF KK(I)<=MIN THEN MIN=KK(I):IMIN=I
150 NEXT I
160 KK1(J)=MIN
170 KK(IMIN)=MAX
180 NEXT J
190 FOR I=1 TO 10
200 PRINT KK1(I)
210 NEXT I
220 DATA <ЗНАЧЕНИЯ ЭЛЕМЕНТОВ>
RUN
```

7.6. Операционная система

Операционная система является неотъемлемой частью программного обеспечения микроЭВМ и, помимо функ-

ций автоматизации разработки программ, реализуемых трансляторами, выполняет упорядочение, планирование и регулирование заданий, программирование ввода-вывода, защиту системы от пользователей, защиту пользователей друг от друга, управление внешней памятью, обработку ошибок.

В настоящее время получили широкое распространение и фактически стандартизованы несколько «семейств» ОС, ориентированных на определенные типы микропроцессоров. Наиболее распространены среди них операционные системы CP/M, MS-DOS, UNIX, RT11, RSX.

Принципы построения операционных систем для микроЭВМ во многом совпадают с принципами построения ОС мини-ЭВМ. Для более детального знакомства и изучения организации ОС рассмотрим операционную систему CP/M — COMMAND PROGRAMM/MONITOR. На базе CP/M построены многие другие операционные системы: MP/M-II, CP/M-86, MP/M-86 и CP/NET.

Базовый состав аппаратуры для ОС включает: процессор с системой команд, соответствующей версии CP/M; ОЗУ; ПЗУ для хранения программ начальной загрузки; дисплей; клавиатуру, внешнюю память на гибких магнитных дисках и печатающее устройство. Отдельные версии ОС допускают подключение более широкого набора периферийных устройств.

Структуру ОС можно представить в виде ядра (резидентной части) и оболочки. Ядро ОС загружается в оперативную память сразу же после включения питания и хранится там все время, пока работает микроЭВМ.

Ядро ОС содержит следующие основные компоненты: файловую систему, драйверы периферийных устройств и командный процессор (интерпретатор командной строки).

Файловая система составляет основу ОС и предполагает организацию хранения, обработки, ввода и вывода данных файлами. *Файл* — именованная совокупность данных, предназначенных для размещения во внешней памяти (в данном случае — на дисках НГМД). В файлах могут содержаться любые сведения: числовая информация и текстовые документы, исходные и объектные программы, таблицы и графики в закодированном виде и пр.

Файл имеет свое обозначение, состоящее из двух частей: имени и типа, разделенных точкой.

Имя файла выбирается произвольно и состоит не

более чем из 8 знаков. Тип (расширение) файла устанавливается в соответствии с характером хранимой информации. Обычно тип назначается операционной системой, состоит из трех знаков и обозначает, например, копию одноименного файла, вспомогательные программы (подпрограммы) файла-программы и др. Ниже приведен пример системных типов файлов.

COM — файл команд (программа).
MAC — исходная программа на МАКРОАССЕМБЛЕРЕ.
REL — объектная программа, полученная при трансляции.
PRN — файл печати, полученный при трансляции исходной программы.
BAK — дублирующий файл.
SUB — файл текста с резидентными или транзитными программами.
BAS — исходная программа на языке БЕЙСИК.

Все файлы дискеты зарегистрированы в справочнике (DIRECTORY). Для выполнения файл-программа считывается с дискеты и помещается в ОЗУ. Файлы обозначаются следующим образом:

COMMAND.COM
START.BAT
UP1.BAK

XONIX.EXE
PROG1.MAC
PLAY-1.BAS

В сокращенном обозначении файла тип может отсутствовать.

Для обозначения группы файлов применяют специальную символику. Например, если необходимо выполнить операцию над всеми файлами с одним именем и различными типами, то вместо типа задается метасимвол «*». Для указания различных имен с одним типом вместо имени также ставится «*». Например, сообщение UM.* обозначает файлы с именем UM и всевозможными типами, а сообщение *.COM обозначает файлы со всевозможными именами типа COM. Метасимвол «?» используется для обозначения всевозможных букв и цифр при записи имени и типа. Например, сообщение UM? обозначает все файлы, имена которых состоят из трех знаков и имеют в своем имени две начальные буквы UM.

Драйверы периферийных устройств осуществляют программную поддержку периферийных устройств, входящих в состав микроЭВМ. Под этим подразумевается опрос готовности устройства к приему и выдаче информации, контроль правильности ввода/вы-

вода, преобразование форматов команд и данных ОС — ПУ и ПУ — ОС и др.

Драйверы стандартных устройств образуют базовую систему ввода-вывода (BIOS — от англ. base input-output system), которая полностью или частично хранится в ПЗУ. При помощи BIOS выполняется начальная загрузка ОС — считывание с НГМД в ОЗУ резидентной части операционной системы после включения питания микроЭВМ.

Командный процессор (ССР — от англ. command console processor) реализует прием инструкции пользователя, ее редактирование и выполнение. ССР — это та часть ОС, которая непосредственно взаимодействует с пользователем и от качества построения которой во многом зависит удобство работы с ОС, простота ее освоения и эксплуатации. Совокупность функций, выполняемых ОС, определяется набором команд ССР.

Команды ССР подразделяют на резидентные, транзитные и команды управляющих символов.

К *резидентным* относятся команды, которые реализуются программами, загружаемыми в ОЗУ при начальной загрузке и постоянно находящимися в оперативной памяти ЭВМ. Резидентные команды не регистрируются в справочнике, поэтому их часто называют также *встроенными*.

Резидентные команды

DIR — листинг справочника на экране дисплея.

ERA — (erase) стирание файлов.

REN — (rename) переименование файлов.

SAVE — сохранение области ОЗУ на дискете НГМД.

TYPE — вывод содержимого файла на печать (дисплей или ПЧУ).

USER — установка зоны пользователя.

Готовность к приему команды операционная система индицирует выводом имени активного дисководов (дисковод, с которым осуществляется взаимодействие) и знака «>». В качестве имени дисководов в CP/M используются обозначения: A: B: C: и т. д., причем системным (главным) дисководом считается накопитель с именем A:. С системного накопителя осуществляется считывание ядра ОС при начальной загрузке. Например, сообщение

```
XXX CP/M VERS Y.Y  
A>
```

является типичным для большинства версий CP/M. Здесь «XXX» представляет собой числовое значение,

равное объему ОЗУ в килобайтах, на использование которого настроена данная версия ОС. К примеру, в версии 1.4 «ХХК» может меняться от 16 К до 64 К. В версии 2.0 и выше это значение изменяется от 20К до 64К. Точное значение зависит от объема памяти микроЭВМ, специфицированной разработчиком или поставщиком ОС.

Вторая часть надписи «VERS Y. Y» указывает версию ОС СР/М: 1.3, 1.4, 2.0, 2.2 и т. д. В зависимости от версии СР/М и ее поставщика в сообщении может содержаться дополнительная информация: реквизиты разработчика, дата генерации системы, список подключенных внешних устройств и т. п.

После появления «А» пользователь может вводить команду ССР. Команда состоит из имени команды и операндов — обозначений файлов, с которыми выполняется команда. Окончание команды ССР индицируется нажатием клавиши <ЕТ>. Если файл расположен не в активном накопителе, то для обозначения его местонахождения используют конструкцию И:ИМЯ.ТИП, где И — ИМЯ накопителя, ИМЯ.ТИП — обозначение файла. Например, файл TLP.DOC, расположенный на диске в дисковом В:, обозначается В:TLP.DOC.

Примеры использования резидентных команд.

Необходимо распечатать каталог файлов дискеты А

В> DIR A:*. * <ЕТ>

или В> DIR A: <ЕТ>

Если в качестве активного в настоящий момент используется дисковод А, то достаточно ввести

А> DIR <ЕТ> .

Стереть файл с обозначением CNB.COM на активном дисковом.

А> ERA CNB.COM <ЕТ>

Если необходимо стереть все файлы с дискеты В, то

А> ERA B:*. * <ЕТ>.

Переименовать файл CNB.COM. Новое имя —

CNB.NEU. Оба файла на диске В.

А> REN B:CNB.COM=B:CNB.NEU <ЕТ>

Вывести на экран дисплея текстовый файл CNB.PRN, расположенный на диске В.

А> TYPE B:CNB.PRN <ЕТ>

Одновременным нажатием клавиш CTRL (от англ. control — управлять) и S индикация прекращается до следующего ввода CTRL/S. Нажатием CTRL/P можно осуществить одновременный вывод на печатающее устройство.

Переписать данные из ОЗУ, начиная с адреса 100H (зона пользователя) на активную дискету в качестве файла с именем BAL.COM. Объем данных - 15 блоков по 256 байт.

A> SAVE 15 BAL.COM <ET>

Установить в качестве активного дисковода с именем B.

A> B: <ET>.

Транзитными или переходными называются команды, которые лишь на время выполнения загружают соответствующие программы в зону оперативной памяти, отведенную для пользователя. Чтобы та или иная команда выполнялась, соответствующий файл с программами должен быть помещен в ВЗУ на дискете и зарегистрирован в каталоге.

Переходные команды

FORMAT — форматирование (разметка) дискеты. С помощью этой команды осуществляется подготовка дискеты к записи совместимых с CP/M файлов.

SYSGEN — генерация операционной системы. Команда записывает резидентные программы ОС на системные магнитные дорожки дискеты. Обычно это дорожки с номерами 0 и 1.

DUMP — индикация файла на дисплее или печатающем устройстве в шестнадцатеричном формате.

PIP — (peripheral interchange program) — обмен данными между периферийными устройствами. С помощью команды можно удалять, копировать, дублировать файлы, создавать цепочку файлов.

STAT — предоставление информации о состоянии системы.

ED — (editor) — редактор. Команда запускает текстовый редактор, позволяющий создавать и редактировать текстовые файлы.

SUBMIT — предоставляет возможность автоматического запуска указанной последовательности команд, предварительно подготовленной пользователем в отдельном файле.

ASM — (assembler) — транслирует файлы-программы с исходным текстом на языке АССЕМБЛЕР.

DDT — (dynamic debugging tool) — инструментальное средство для динамической отладки программ. Команда позволяет определять и устранять ошибки в объектных программах.

Набор транзитных команд может быть расширен или сужен в зависимости от версии ОС и потребностей пользователя.

Примеры использования переходных (транзитных) команд.

Вывести в шестнадцатеричном формате на экран дисплея содержимое файла PRG1.COM.

A) DUMP PRG1.COM <ET>

Переписать файл CNB.BAS с дискеты В на дискету А.

A) PIP <ET>

* A:=B:CNB.BAS <ET>

Скопировать все файлы дискеты А на дискету В.

A) PIP <ET>

* B:=A:*. * <ET>

Вывести объем файла PRG2.COM в килобайтах.

A) STAT PRG2.COM <ET> .

Команды управляющих символов предоставляют пользователю возможность редактировать командную строку, прерывать выполнение некоторых команд ССР, реинициализировать ОС. Эти команды выполняются одновременным нажатием управляющей клавиши CTRL и какой-либо другой буквенной клавиши. Например, нажатие клавиши CTRL/C приводит к появлению на экране дисплея сообщения ^C и осуществляет реинициализацию (перезапуск) ОС.

Некоторые команды для удобства совмещены с управляющими клавишами маркера.

Оболочка ОС — это набор системных программ, расширяющих функции ядра операционной системы. К ним относятся трансляторы, редакторы, библиотеки и др.

7.7. Пакеты прикладных программ

Пакет прикладных программ может иметь простую и сложную структурную организацию.

Простая структурная организация ППП предполагает набор независимых программных модулей, не связанных между собой. Для работы с ППП пользователь должен написать программу, вызывающую нужный модуль из ППП и выполняющую необходимые действия. Простую структуру обычно имеют пакеты, ориентированные на расширение вычислительных функций. Пользователь должен знать состав пакета и правила обращения к его программным модулям.

ППП со сложной структурной организацией, помимо основных обрабатывающих программ, включают и ряд управляющих и сервисных модулей. Работа с такими

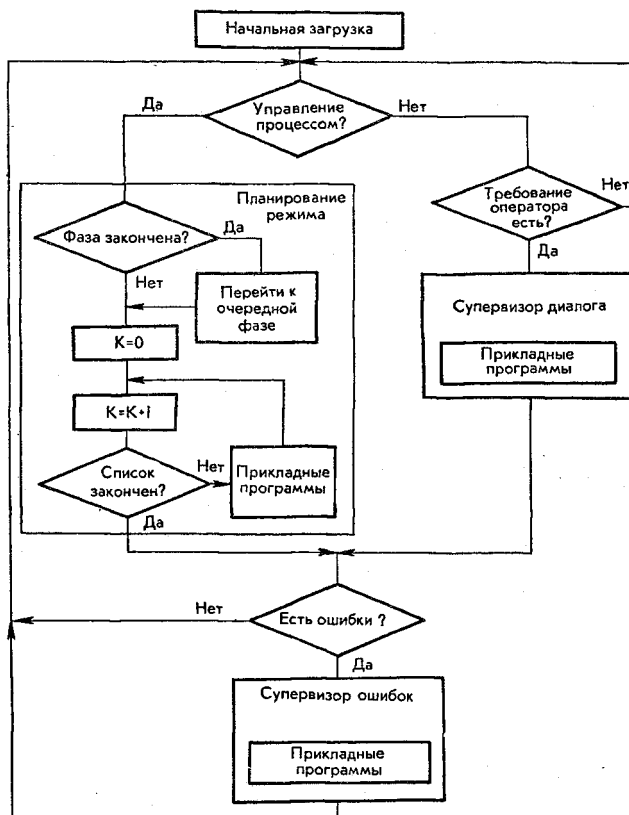


Рис. 7.15

на базе микроЭВМ «Электроника-60М» и состоит из процессора, ОЗУ 16К×16, ПЗУ 12К×16, устройства отображения, клавиатуры, специализированного процессора формирования и анализа гармонических сигналов (ПГС) и специализированного процессора формирования и анализа широкополосных случайных сигналов (ПСС). Указанный ППП относится к классу пакетов со сложной структурной организацией и включает следующие модули: главный планировщик, планировщик режима испытаний, супервизор диалога (СД), супервизоры связей, супервизоры прерываний, супервизоры ошибок.

Основной задачей *главного планировщика* (рис. 7.15) является планирование работ в системе, поскольку запро-

сы к ней недетерминированы и зависят от многих факторов: требований оператора, готовности данных к передаче исполнительному устройству, прерываний по таймеру и т. п. Управление главному планировщику может быть передано сразу же после включения питания микроЭВМ. Всякая программа, инициированная главным планировщиком, организует обратную передачу управления независимо от результата ее выполнения.

В структуре функций главного планировщика следует выделить два раздела:

планирование работ в статике, когда выполняются программы настройки системы на требуемый вид испытаний;

планирование работ в динамике, когда организуется управление реальным процессом.

Инициирование программных модулей первого раздела происходит по требованию оператора, например ввод параметров, задание графиков, масштабирование, аппроксимация двух точек графика и т. д. После нормального выполнения запроса оператора главный планировщик устанавливается в режим ожидания следующего запроса. Все остальные запросы в первом разделе блокируются. Это является дополнительной мерой устранения сбойных и неоднозначных ситуаций.

Очередность выполнения программ во втором разделе определяется *планировщиком режима* в соответствии со следующими правилами.

Процесс управления экспериментом разбивается на крупные этапы (фазы), объединяющие совокупность одинаковых по своему целевому назначению работ, а каждая фаза в свою очередь разделяется на некоторое конечное число K_m шагов. Имя совокупности прикладных программ, выполняемых в I -й фазе ($I=0, 1, 2, \dots, I, \dots, M$; здесь $M+1$ — количество фаз), заносится в I -ю строку списка фаз. Под этим именем в ПЗУ хранится список шагов, структура которого аналогична структуре списка фаз: в K -й строке ($K=0, 1, 2, \dots, K_m$) содержится имя модуля, выполняющего K -й шаг. Вначале планировщик режима задается параметрами $I=0, K=0$ и организует пошаговое выполнение I -й фазы. После первого цикла выполнения программных модулей I -й фазы соответствующей прикладной программой устанавливается ключ окончания фазы. Если этого не произошло, то планировщик режима организует повторение выполнения I -й фазы до тех пор, пока ключ не установится. Затем

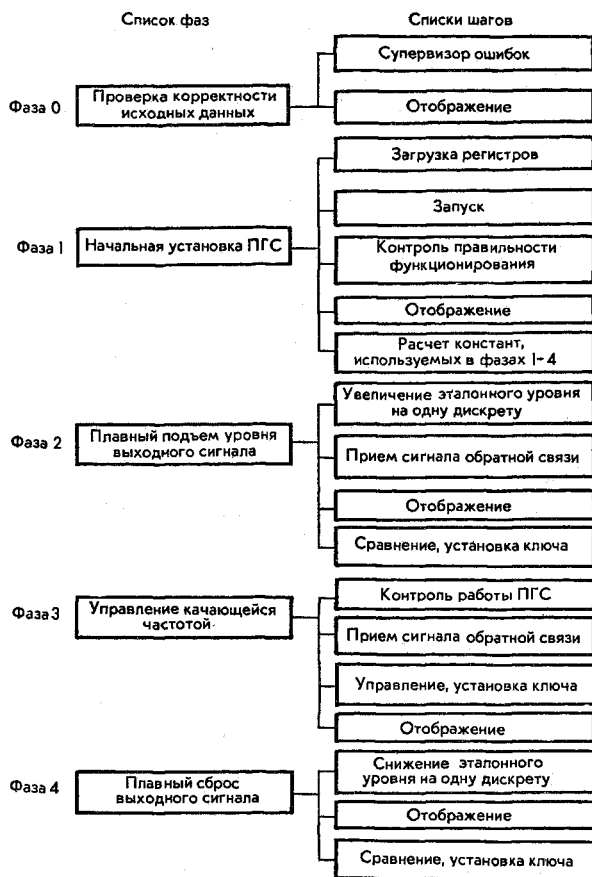


Рис. 7.16

инкрементируется $I = I + 1$ и выполняется очередная фаза. Условие окончания I -й и перехода к выполнению $(I + 1)$ -й фазы определяется конкретными алгоритмами работы фаз, характером и ходом эксперимента.

Основные принципы функционирования планировщика режима иллюстрируются рис. 7.16, где приведен пример подготовки списков для управления гармонической вибрацией с качающейся частотой. Списки составляются с учетом специфики проводимого эксперимента самим инженером-испытателем, использующим готовые модули, либо программистом и помещаются в ОЗУ или

ПЗУ в зависимости от степени дальнейшей переработки списков.

Такая структура планирования работ обеспечивает гибкость и быструю адаптацию алгоритма управления экспериментом, а также дает возможность пользователю самому внести изменения в этот алгоритм. На каждой фазе испытаний осуществляется обращение к графическим программам для отображения информации. Это является одной из важных отличительных особенностей ПО, позволяющей системе организовать непрерывную связь с оператором, информировать не только о конечном результате или ошибочной ситуации, но и выдавать сообщения о промежуточных действиях, обеспечивая тем самым визуальный контроль за проведением испытаний в интерактивном режиме.

Выделение программ управления диалогом в отдельный модуль СД дает такие преимущества, как независимость ПО от конкретных форм ведения диалога, предоставление пользователю права самому программировать диалог путем изменения только одного супервизора, широкие резервы для совершенствования диалога без изменения остальных модулей.

Диалог в режиме настройки предполагает взаимодействие по инициативе оператора. При этом расположение необходимой информации, компоновка и содержание сообщений определяются супервизором в соответствии с характером требуемого запроса. Каждый запрос заканчивается отображением на экране определенной информации, сигнализирующей об окончании выполнения действий, например ввод параметров: частоты, мощности сигнала, чувствительности датчика и т. п.

В режиме управления экспериментом планировщиком режима организуется выдача сообщений о ходе эксперимента, а также обработка запросов оператора. При этом количество возможных запросов существенно ограничивается, сохраняются лишь самые необходимые, а за счет высвобожденного времени обработки запросов может быть повышена информативность сообщений.

Связь программ с аппаратными средствами и реализованная в них программными или микропрограммными модулями обеспечивается набором различных *супервизоров связей*. Каждый супервизор имеет свою конкретную структуру, определяемую его назначением.

Супервизоры прерываний управляют обработкой информации при возникновении события, обусловленного

внешними по отношению к ППП факторами. Основные задачи супервизоров прерываний следующие: сохранение состояния прерванной программы; выполнение действий, связанных с обработкой прерывания; восстановление состояния прерванной программы.

Супервизоры ошибок предназначены для выявления и, если возможно, устранения неправильных действий оператора, а также исключения недопустимых значений параметров.

Графические, служебные, сервисные и прикладные программы образуют простой пакет, состоящий из набора модулей, реализующих определенные функции. Примерами могут служить программы преобразования кодов КОИ-7 в число с плавающей запятой, вывода сообщения-подсказки оператору, расчета по математическим формулам, используемым при работе ППП и др.

7.8. Технология программирования

По мере увеличения размеров составляемых программ растет и сложность процесса программирования. Небольшие программы, составленные одним человеком, не требуют, как правило, сложной технологии программирования. Программу пишут на одном из языков высокого уровня с использованием библиотеки стандартных подпрограмм, транслируют, отлаживают и запускают в работу.

Важным этапом в общем технологическом цикле является отладка программы — поиск и устранение ошибок, которые возникли при составлении программы.

Статистические данные, приведенные в литературе, показывают, что хорошо подготовленный программист в процессе создания ПО может допустить до пяти ошибок на 100—120 предписаний, причем при исправлении программы иногда допускаются новые ошибки, т. е. процедура устранения ошибок в ПО является итерационной. Некоторые из ошибок обнаруживаются только при такой комбинации исходных данных, какую трудно предусмотреть в процессе отладки и которая может встретиться лишь через несколько лет нормальной эксплуатации программы. Ярким примером тому является ПО космических полетов APOLLOS. По мнению специалистов оно является наиболее проверенным в мире. Однако, несмотря на это, во время полетов APOLLO-8, APOLLO-11, APOLLO-14 были обнаружены ошибки ПО.

Все множество ошибок, имеющих место в комплексах алгоритмов и программ микроЭВМ, обычно классифицируют таким образом:

системные, связанные с неправильным пониманием требований задачи и условий ее реализации;

алгоритмические, вызванные некорректной формулировкой и реализацией алгоритмов программным путем;

программные, обусловленные неправильностью записи программы (описки, логические ошибки, ошибки кодирования, адресации и т. д.);

технологические, возникающие в процессах подготовки документации на программу и перевода ее на машинные носители.

Отладка программы предусматривает использование специальных тестов (проверок) для всей программы или ее отдельных блоков. В простейшем случае — это набор исходных данных, для которых результат очевиден или легко вычисляется без ЭВМ. Например, для вычисления по формуле

$$y = (1 + 1/x)^x$$

в качестве x могут быть взяты целые числа 1, 2, 3.

Для поиска ошибок может использоваться так называемый покомандный (или поблочный) режим работы, при котором после выполнения очередной команды (или блока команд) происходит останов ЭВМ. После этого программист может вывести содержимое интересующих его ячеек памяти, проанализировать состояние программы и оценить промежуточные результаты.

В более сложных системах тестирования оператору-отладчику выдается сообщение о содержимом всех используемых ячеек памяти, состоянии процессора и др.

Многие программные ошибки выявляются транслятором на этапе лексического и синтаксического анализа программы. Однако следует помнить, что транслятор осуществляет лишь формальный контроль правильности использования символов и конструкции операторов. Что же касается содержательных ошибок программы (семантического контроля), то их выявление и исправление требуют только совместной работы ЭВМ и программиста. К таким содержательным ошибкам относятся, например, неправильный переход в программе, нарушение требуемой последовательности выполнения операторов, неправильно заданные типы данных и т. п.

Для создания сложных программ, требующих работы

коллектива программистов, необходима более сложная технология программирования. Основное отличие ее от простой технологии состоит в наиболее эффективном распределении работы по написанию отдельных частей программы между программистами и осуществлении последующей стыковки этих частей.

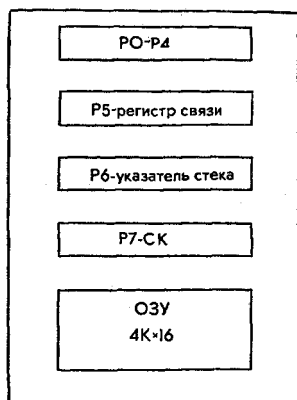


Рис. 7.17

С этой целью на этапе формализации и первичной алгоритмизации процессов, которые необходимо запрограммировать, создают модель микроЭВМ для программиста (программную модель), отражающую взаимосвязь между аппаратными и программными средствами: состав и назначение функциональных модулей, доступных программисту, программную организацию работы аппаратных средств, размещение программных модулей в системе.

ме, динамику перемещения их в процессе вычислений и др. Чем больше возможностей предоставляется программисту, тем более подробная и полная модель требуется ему.

Примером полной программной модели может служить рис. 7.17, на котором приводятся основные элементы, доступные программисту в процессоре микроЭВМ «Электроника-60М». В качестве более сложного примера рассмотрим программную модель автоматизированной системы управления вибрационными испытаниями АСУ ВИ (рис. 7.18). Модель включает аппаратные средства системы (выделены штриховыми линиями), связи между ними (штрихпунктирные линии), программные модули и структуры данных, «привязанные» к месту их размещения, связи между этими модулями.

После разработки программной модели создается обобщенная граф-схема процесса вычислений, в которой указываются последовательность обработки данных, условия входа в ту или иную программу и выхода из нее, оговаривается интерфейс между программными модулями и четко определяются их функции.

Затем написание отдельных программ может быть поручено разным программистам.

Такой процесс получил название нисходящего проектирования, или проектирования сверху вниз. Другим типом технологического процесса создания программ является восходящее проектирование, или проектирование снизу вверх, при котором вначале

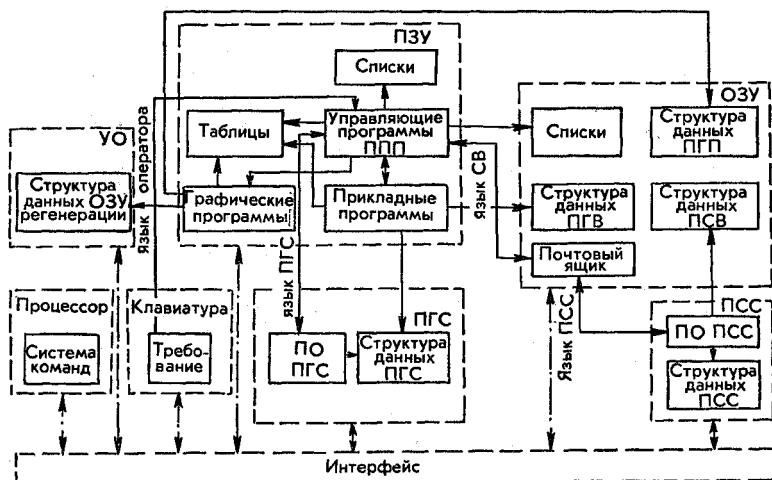


Рис. 7.18

создаются наборы программных модулей, решающих отдельные задачи, а далее из этих модулей компонуется полная программа.

Процесс создания программ является итерационным. Это означает, что на каждом этапе разработки программы возможен возврат к предыдущим этапам с целью корректировки программы или сопровождающих ее документов. Например, обнаружив ошибку при исполнении программы, оператор должен скорректировать исходный модуль и повторить трансляцию, редактирование и выполнение программы.

В сложных технологиях программирования возврат к предыдущим этапам в общем случае предполагает повторение тех или иных действий всего коллектива программистов. Поэтому особое внимание уделяется начальным этапам процесса проектирования, на которых закладываются принципы построения программ.

Контрольные вопросы и задания

1. Что называется программным обеспечением ЭВМ?
2. Из каких частей состоит программное обеспечение?
3. В чем различие между операционной системой, пакетами прикладных программ и программами диагностики и технического обслуживания?
4. Какими преимуществами и недостатками обладают машинно-ориентированные языки?
5. Каковы достоинства и недостатки машинно-независимых языков?
6. Какие типы трансляторов вы знаете? Чем они характеризуются?
7. Перечислите этапы прохождения задачи через ЭВМ?
8. Чем характеризуется программная модель микроЭВМ?
9. Какие основные компоненты содержит операционная система?
10. Как обозначаются файлы в операционной системе CP/M?
11. Какие функции операционной системы реализуются резидентными, а какие — переходными командами?
12. В чем различие языков АССЕМБЛЕР и МАКРОАССЕМБЛЕР?
13. Какие средства МАКРОАССЕМБЛЕРА приближают его к языкам высокого уровня?
14. Чем различаются режимы непосредственного счета и программный в языке БЕЙСИК?
15. Какие функции выполняются директивами языка БЕЙСИК?
16. Какими операторами реализуются условный и безусловный переходы в языке БЕЙСИК?
17. Перечислите операторы ввода и вывода данных в языке БЕЙСИК.
18. Назовите операторы, реализующие циклы в языке БЕЙСИК.
19. Какие типы констант и переменных допускаются в языке БЕЙСИК?
20. Чем различаются ППП с простой и сложной структурной организацией?

Упражнения

1. Напишите в машинных кодах системы команд DEC программы обнуления ячеек памяти 100—200, получения из четырех цифр в коде КОИ-7 двоично-десятичного числа (исходные коды и результат — в памяти ЭВМ), сложения четырех чисел с фиксированной запятой.
2. Напишите на языке МАКРОАССЕМБЛЕР программу организации в памяти ЭВМ таймера, индицирующего текущее время в часах, минутах и секундах.
3. Напишите на языке МАКРОАССЕМБЛЕР программу вывода на печать символа в коде КОИ-7. Регистры состояния и данных ПЧУ соответственно имеют адреса 177600 и 177602. Флаг готовности устанавливается в 7-м разряде регистра состояния.
4. Напишите на языке МАКРОАССЕМБЛЕР программу формирования пилообразного напряжения 0—10,2 В. Состав аппаратуры: микроЭВМ «Электроника-60М» с модулем ЦАП, имеющим регистр состояния с адресом 160000 и регистр данных — 160002, куда в прямом коде заносится цифровое значение формируемого напряжения. Единица младшего двоичного разряда соответствует напряжению 10 мВ, флаг готовности установки напряжения устанавливается в 7-м разряде регистра состояния.

5. Напишите на языке БЕЙСИК программу поиска элементов массива $A(I)$, удовлетворяющих требованиям $X < A(I) < Y$.

6. Напишите на языке БЕЙСИК программу расстановки заданных слов по алфавиту.

7. Напишите на языке БЕЙСИК программу решения уравнения $ax^2 + bx + c = 0$.

8. Напишите на языке БЕЙСИК программу построения графика функции $y = \cos(x)$. График можно выводить символами «*» с осями координат, развернутыми на 90° .

Глава 8. ПРИМЕНЕНИЕ МИКРОЭВМ В РОБОТОТЕХНИКЕ И СИСТЕМАХ АВТОМАТИЗАЦИИ ИСПЫТАНИЙ

8.1. Организация микропроцессорных систем управления роботами

Системы управления промышленными роботами (ПР) предназначены для решения следующих задач:

- управления последовательностью технологических операций;

- управления отдельными операциями;

- управления взаимодействием с внешней средой.

Управление последовательностью операций связано с определением порядка следования операций в цикле, порядка чередования циклов, синхронизацией действий робота и обслуживаемого им оборудования.

Управление операциями включает обеспечение требуемых законов движения рабочего органа и режимов работы технологического оборудования. При этом решаются задачи расчета координат, скоростей и ускорений звеньев, перемещения схвата в определенные точки или по определенному контуру, захватывания и отпускания предметов, их ориентации, формирования технологических команд.

Управление взаимодействием со средой строится на основе анализа информации, поступающей от датчиков очувствления и адаптации, систем управления верхнего уровня, других роботов, а также от оператора. Оно заключается в изменении последовательности и характера операций при изменении условий функционирования робота.

Применяемые в настоящее время системы управления промышленными роботами можно условно разделить на цикловые, позиционные, контурные и позиционно-контурные.

Цикловые системы обеспечивают разомкнутое управление исполнительными приводами (типа «включено-выключено») и ориентированы на работу с двухпозиционными датчиками (например, конечными выключателями). Они используются при автоматизации дискретных технологических процессов с небольшим числом

многократно повторяющихся операций (штамповка, пресование, термообработка и др.). При таком управлении возможно позиционирование манипулятора лишь в крайних положениях (по упорам), а взаимодействие с внешней средой ограничено вводом информации с пульта и обменом сигналами с технологическим оборудованием о готовности к началу операции, ее окончании, аварии и т. д. Программирование цикловых систем производится путем настройки механических упоров и записи циклограммы на программноносителе.

В позиционных системах реализуется замкнутое управление исполнительными приводами, для чего каждое звено снабжается датчиком обратной связи. Однако позиционирование возможно лишь в отдельных точках рабочего пространства, координаты которых задаются при программировании. Общее число этих точек ограничивается объемом памяти системы управления. Уставки приводов изменяются скачкообразно, после отработки очередной позиции; траектория движения схвата от точки к точке не контролируется. Позиционные системы используются при автоматизации процессов точечной сварки, обслуживании металлорежущих станков. Они программируются в режиме обучения либо с применением языков высокого уровня.

В контурных системах также используется замкнутое управление исполнительными приводами, но уставки изменяются непрерывно во времени. В результате обеспечивается плавное перемещение исполнительного органа по требуемой траектории с заданными скоростью и ускорением. Обычно в контурных системах запоминаются лишь узловые точки траектории, а координаты промежуточных точек определяются в результате интерполяции (линейной, круговой, параболической). Область применения контурных систем — автоматизации процессов окраски и дуговой сварки; их программирование осуществляется путем обучения или с помощью языков высокого уровня.

Позиционно-контурные системы характеризуются сочетанием элементов движения как позиционных, так и контурных систем. В них реализуется последовательная отработка большого числа точек, лежащих на заданной траектории.

При управлении манипуляторами с простой кинематической схемой и небольшим числом степеней подвижности система управления робота строится на базе одной

микроЭВМ. Типичная схема такой системы (рис. 8.1) содержит 8- или 16-разрядный процессор, ОЗУ, ПЗУ, ВЗУ на кассетной магнитной ленте или гибком магнитном диске, интерфейсы пультов управления и обучения, контроллер приводов, модуль ввода-вывода дискретных сигналов и команд, модуль связи с ЭВМ верхнего уровня.

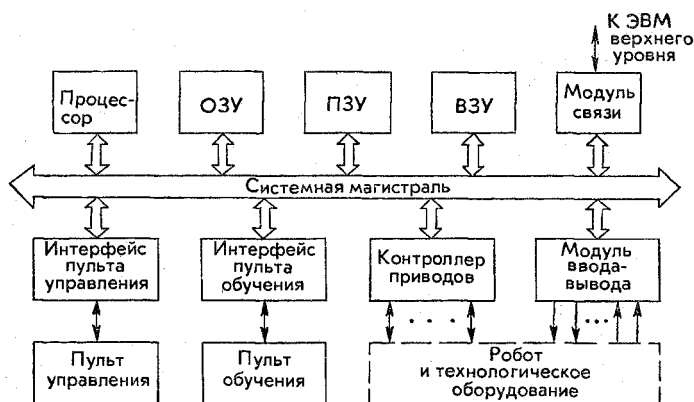


Рис. 8.1

Процессор работает в режиме разделения времени, обеспечивая последовательно выполнение нескольких программ — планирования траектории робота, преобразования координат, интерполяции, управления исполнительными приводами и др. Выполнение этих программ может прерываться по сигналам от датчиков робота и технологического оборудования. Программа движения робота обычно записывается на магнитный носитель или в перепрограммируемое ПЗУ.

Для управления манипуляторами со сложной кинематической схемой и большим числом степеней подвижности используются многопроцессорные системы. В таких системах главный (центральный) процессор решает задачи общего характера, обеспечивая планирование движений, преобразование координат, интерполяцию в декартовом пространстве, связь с внешней средой, обучение и программирование робота. Подчиненные процессоры осуществляют управление отдельными степенями подвижности. Иногда функции планирования траектории, преобразования координат и обработки технологической информации выполняются специализированными процессорами.

При построении многопроцессорных систем управления роботами наибольшее распространение получили две архитектуры — с общей шиной и общей памятью.

Системы с общей шиной имеют иерархическую структуру (рис. 8.2), в которой выделяются системная шина центрального процессора, шина связи и местные шины.

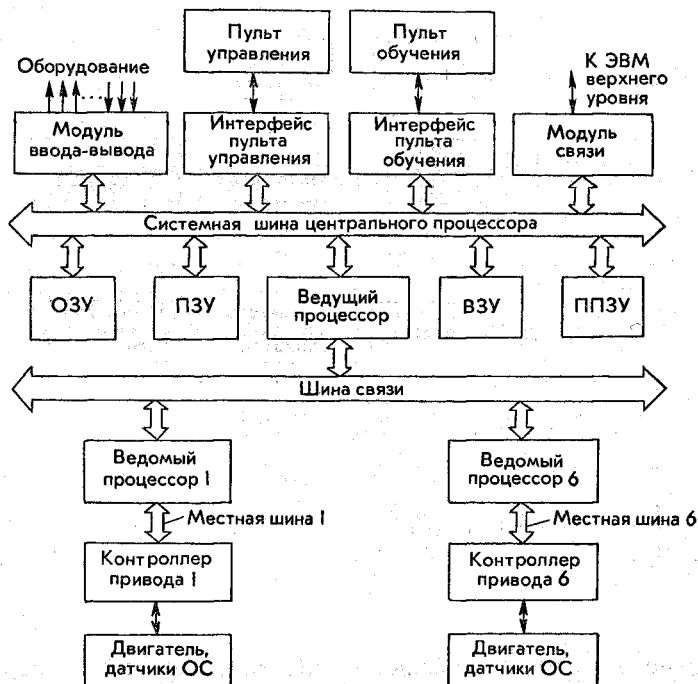


Рис. 8.2

Центральная системная шина служит для обмена данными между ведущим процессором и внешними функциональными узлами. Шина связи обеспечивает параллельную работу ведомых процессоров и обмен данными между иерархическими уровнями. Местные шины используются для сопряжения ведомых процессоров с модулями управления исполнительными приводами.

Системы с общей памятью базируются на нескольких микроЭВМ, соединенных одной общей шиной (рис. 8.3). Обмен данными между ними осуществляется не на пря-

мую, а через специально выделенные области памяти. Общая память организуется таким образом, что запись в любую ячейку может производиться только одним вычислительным модулем, а чтение данных из ячейки — всеми модулями.

Недостатком систем с общей шиной является то, что обмен информацией возможен только через центральный

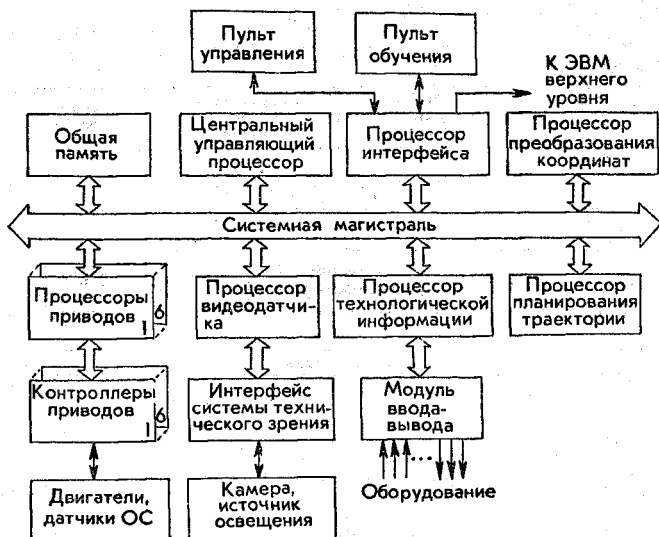


Рис. 8.3

процессор или с разрешения центрального процессора, а недостаток систем с общей памятью — необходимость двойной передачи данных («источник — общая память — приемник»). Достоинство каждого способа построения проявляется при конкретной реализации и во многом определяется используемой элементной базой.

Многопроцессорные системы управления ПР отличаются также способом реализации алгоритмов управления исполнительными приводами. В простейшем случае все функции управления приводом выполняются аппаратными средствами контроллера, а микропроцессор осуществляет лишь формирование задающего воздействия. В более совершенных системах на микропроцессор возлагаются функции непосредственного цифрового управления приводом, тестирования его элементов.

Программное обеспечение микропроцессорных систем управления роботами включает в себя операционную систему реального времени, язык управления роботом, а также системные, сервисные и функциональные программы.

Операционная система является ядром программного обеспечения. Она представляет собой совокупность программ, обеспечивающих функционирование отдельных элементов системы управления как единого целого. В функции операционной системы входят управление выполнением программ, автоматизация разработки и отладки программных средств, организация взаимодействия программ друг с другом, внешними устройствами, датчиками и исполнительными механизмами.

Язык управления роботом состоит из двух компонентов — языка программирования и языка директив (команд). Язык программирования робота по своим функциям подобен языку программирования универсальной ЭВМ, но содержит ряд специфических операторов — перемещения руки робота, открытия и закрытия схвата, опроса датчиков, временной задержки и т. д. Наиболее развитые языки программирования роботов содержат также операторы синхронизации и параллельного выполнения задач.

Язык директив предназначен для организации диалоговой связи человека-оператора с роботом. Он позволяет установить режим функционирования робототехнического комплекса, задать начальную конфигурацию манипулятора, инициировать выполнение задачи, приостановить робот в критических ситуациях, получить сведения о наличии свободной памяти и т. д. Предложения языка директив называются командами.

В состав *системных и сервисных программ* робота обычно входят системный загрузчик, компилятор или интерпретатор языка программирования, компоновщик транслированных программ, отладчик, программы организации файлов и доступа к файлам.

Функциональные программы предназначены для реализации конкретных технологических задач и разрабатываются, как правило, пользователями. Иногда в состав программного обеспечения робота включается несколько готовых демонстрационных программ, позволяющих проверить работу робота на некоторых стандартных операциях.

При создании программных и аппаратных средств

систем управления роботами широко используется модульный подход, позволяющий снизить расходы на разработку, производство и эксплуатацию роботов, существенно упростить их обслуживание. В последнее время наметилась тенденция к построению систем управления роботами на базе локальных вычислительных сетей.

Отечественной промышленностью выпускается ряд микропроцессорных систем управления роботами (УКМ-772, «Сфера», «Прогресс», «Гранит» и др.), отличающихся техническими характеристиками, назначением и архитектурой.

8.2. Система управления «Гранит-02»

Система управления «Гранит-02» предназначена для циклового управления роботами и оборудованием, используемым при автоматизации технологических процессов дискретного типа (штамповка, прессование, литье под давлением и др.). Она рассчитана на подключение двухпозиционных исполнительных механизмов и датчиков, работающих по принципу «включено-выключено».

Система реализована на базе 8-разрядного микропроцессора КР580ИК80А и обеспечивает функционирование робота и оборудования в автоматическом, полуавтоматическом, шаговом и ручном режимах. Программные средства системы позволяют пользователю разрабатывать технологические программы на проблемно-ориентированном языке, редактировать их и отлаживать непосредственно на управляемом объекте. Связь системы с ЭВМ верхнего уровня осуществляется по интерфейсу ИРПС.

Техническая характеристика системы «Гранит-02»

Элементная база	МПК КР580
Разрядность микропроцессора	8
Объем ОЗУ, байт	256
Объем ПЗУ, К байт	8
Объем ППЗУ пользователя, К байт	4
Количество каналов ввода	64
Количество каналов вывода	64
Время реакции на изменение состояния объекта, мс	Не более 50

Структурная схема системы управления «Гранит-02» приведена на рис. 8.4. Система построена по блочно-мо-

дульному принципу и состоит из блока микроЭВМ, блока ввода-вывода, пультовой аппаратуры и блока питания.

Блок микроЭВМ включает в себя модуль процессора, ПЗУ, ППЗУ (модуль памяти пользователя), модуль связи с ЭВМ, адаптер. *Модуль процессора* является ведущим и производит все необходимые операции по

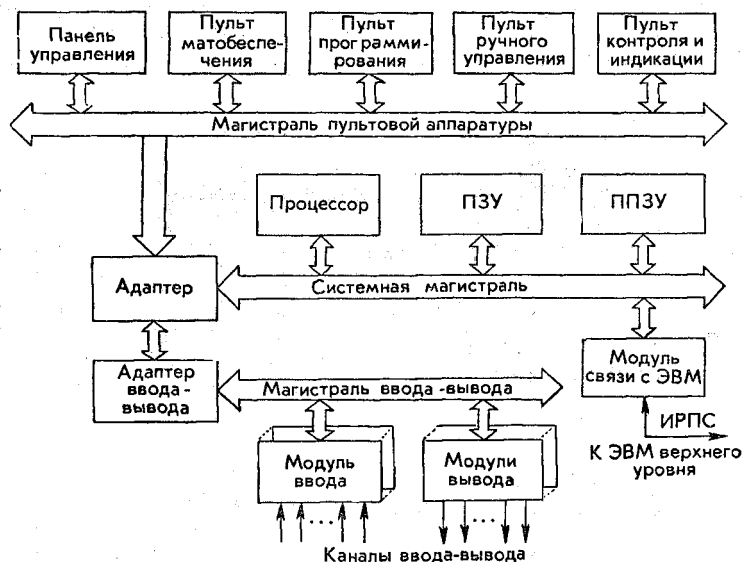


Рис. 8.4

арифметической и логической обработке информации. Для организации вычислительного процесса и хранения промежуточных результатов модуль процессора содержит также резидентное ОЗУ объемом 256 байт. *Модуль ПЗУ* предназначен для хранения системного программного обеспечения, *модуль ППЗУ* — для хранения технологических программ. Причем в ППЗУ хранится как сама технологическая программа, так и ее дубль и контрольная сумма. *Модуль связи* и *адаптер* служат для организации информационного обмена с внешней ЭВМ, блоком ввода-вывода и пультовой аппаратурой.

Блок ввода-вывода состоит из адаптера и модулей ввода-вывода. *Адаптер* ввода-вывода выполняет частичную дешифрацию адресов и буферизацию сигналов. *Модули ввода* осуществляют прием сигналов от

технологического оборудования и их нормализацию до логических уровней. *Модули вывода* обеспечивают коммутацию цепей управления оборудованием и гальваническую развязку. Применяемые в системе модули рассчитаны на прием и выдачу сигналов со следующими параметрами:

Напряжение входного сигнала, В	= 24, ~24*
Входной ток, мА	20
Напряжение выходного сигнала, В	= 12—48 ~36—110
Выходной ток, А	3

К пультовой аппаратуре относятся панель управления и четыре пульта: программирования, математического обеспечения, ручного управления, контроля и индикации. На *панели управления* размещаются кнопки включения и выключения питания, задания режима работы системы, управления технологическим циклом и средства визуального контроля за ходом выполнения программы. *Пульты* предназначены для ввода и редактирования технологической программы, ремонта, наладки и обслуживания системы, управления каналами вывода в ручном режиме, индикации состояния каналов в процессе работы. Пульты представляют собой конструктивно законченные устройства и подключаются к системе при помощи кабелей.

Блок питания обеспечивает систему рядом стабилизированных напряжений и защищает микроЭВМ от сетевых помех. Кроме того, в модуле ППЗУ предусмотрено резервное питание (подпитка от батарей), позволяющее сохранить записанную информацию при отключении основного питания или изъятии модуля.

Программное обеспечение системы управления «Гранит-02» включает в себя монитор, редактор входного языка, интерпретатор и набор тестов. Монитор имеет аппаратную поддержку в виде пульта математического обеспечения, что позволяет выполнять чтение и модификацию содержимого памяти, запуск и отладку программ в машинных кодах, перемещение содержимого одной области памяти в другую.

Технологическая программа составляется и редактируется с помощью пульта программирования. При этом пользователю доступны следующие виртуальные ресурсы: 64 одноразрядных канала ввода; 64 одноразрядных канала вывода; 8 байтов ввода; 8 байтов вывода;

* — постоянное напряжение, ~ — переменное.

32 таймера; 16 счетчиков; 64 одноразрядных регистра; 1 байт-уставка.

Каналам ввода-вывода присвоены номера от 001 до 064. Группы из 8 каналов могут также объединяться в байты (каналы 001—008 образуют байт 1, каналы 009—016 — байт 2 и т. д.). При этом одни и те же каналы могут участвовать как в битовых, так и в байтовых операциях.

Таймеры имеют номера от 01 до 32 и используются для реализации операций задержки. Они могут быть запрограммированы на время от 0,1 до 25,5 с с дискретностью 0,1 с (таймеры 01—16) или же время от 1 до 255 с с дискретностью 1 с (таймеры 17—32).

Однобитовые регистры (флаги) предназначены для синхронизации и управления режимами работы системы и технологического оборудования. Семь из них (с номерами 01—07) используются для организации автоматического, полуавтоматического, шагового и других режимов работы системы управления, а остальные (с номерами 08—64) находятся в распоряжении программиста.

Счетчики позволяют определить количество повторений отдельных участков программы или внешних событий. Они нумеруются числами от 01 до 16, а их содержимое не может превышать 255.

Байт-уставка представляет собой восьмиразрядное двоичное число, используемое в операциях по обработке и пересылке данных.

Программа может содержать до 999 пронумерованных строк, в каждой из которых располагается один оператор. Все основные операторы делятся на четыре группы: Е (если), Н (надо), П (переход) и У (управление). Ресурсы системы обозначаются буквами: Г — таймер, Р — регистр, С — счетчик, Б — байт-уставка или байт ввода-вывода.

Оператор типа Е — это опрос какого-либо из ресурсов на соответствие определенному состоянию. Например:

Е018,1 — если канал ввода 018 включен (состояние 1);

Е025,0 — если канал ввода 025 выключен (состояние 0);

Н Е05,1 — если таймер досчитал (до значения уставки).

Оператор типа Н — это выполнение каких-либо действий над ресурсами системы. Например:

Н015,1 — включить канал вывода 015;

Н005,0 — выключить канал вывода 005;

НГ03,1 — включить таймер 03.

Оператор типа П — переход на определенную строку программы (условный или безусловный). В случае условного перехода оператору типа П должен предшествовать оператор опроса Е, причем переход осуществляется, если оператор Е не выполняется. При безусловном переходе оператору П может предшествовать любой оператор, отличный от Е. Например, последовательность операторов Е048,1 и П059 интерпретируется как условный переход на строку 059 при выключенном канале ввода 048, а последовательность операторов Н010,0 и П200 — как выключение канала вывода 10 и безусловный переход на строку 200.

Операторы типа У передают управление системным подпрограммам, например У003 — выключение всех каналов вывода, У005 — останов программы, У107 — передача по каналу связи сообщения о режиме работы системы.

Кроме перечисленных операторов, в программе могут использоваться пустой оператор (-0-) и метка начала программной секции (...). Конкретно уставки таймеров и счетчиков задаются вне тела программы.

При наборе операторов с пульта каждый из них транслируется в двухбайтный код. Это сокращает затраты памяти для хранения технологической программы и ускоряет ее обработку интерпретатором. При просмотре и редактировании программы операторы отображаются на пульте в символьном виде.

Для иллюстрации использования операторов приведем типовые фрагменты технологической программы.

```

001 ...      ; начало секции
002 У003      ; выключить все каналы вывода

...
100 Е010,1    ; если канал ввода 10 включен,
101 П104
102 Н005,1    ; то включить канал вывода 005,
103 П105
104 Н015,1    ; иначе включить канал вывода 015
105 -0-

...
200 Е010,1    ; если канал ввода 010 включен,
201 Е020,0    ; а канал ввода 020 выключен,
202 Н015,1    ; то включить канал вывода 015

...
300 Н025,1    ; включить канал вывода 025
301 НГ02,1    ; включить таймер 02
302 ЕГ02,1    ; если таймер досчитал до уставки,
303 Н025,0    ; то выключить канал вывода 025

```

400 Y001 ; инкремент счетчика циклов
401 P1002 ; повторить программу

Технологическая программа может содержать до 16 секций, каждая из которых описывает алгоритм работы отдельного механизма или устройства. Секция делится на кадры, представляющие собой фрагменты программы, начинающиеся с операторов Е и заканчивающиеся оператором Н. Кадр, стоящий в начале программной секции, может начинаться с любого оператора. Поскольку все механизмы комплекса работают по замкнутому циклу, каждая программная секция заканчивается оператором перехода на ее начало.

При выполнении программы интерпретатор последовательно обрабатывает по одному кадру каждой программной секции (сначала первый кадр первой секции; затем первый кадр второй секции и т. д. до последней; после чего снова возвращается к первой секции). Обработка кадра начинается с анализа условий, задаваемых операторами Е. Если эти условия выполняются, то выполняется и исполнительная часть кадра (операторы Н, П, У). В противном случае кадр считается невыполненным и его обработка откладывается до следующего цикла сканирования. Время сканирования не превышает 50 мс. В результате достигается параллельность выполнения программных секций. При необходимости синхронизировать выполнение отдельных кадров используются одноканальные регистры и сигналы внешних датчиков.

Таким образом, аппаратные и программные средства системы управления «Гранит-02» позволяют автоматизировать процесс управления оборудованием с циклическим характером работы. Они нашли применение в гибких производственных модулях порошковой металлургии, робототехнологических комплексах сборки магнитоэлектрических приборов, в комплексах литья пластмассовых изделий и других производствах.

8.3. Система управления «Сфера-36»

Система управления «Сфера-36» предназначена для позиционно-контурного управления многозвенным электромеханическим манипулятором и дискретного управления дополнительным оборудованием, входящим в состав гибкого производственного модуля. Она рассчитана на

непосредственное сопряжение с электродвигателями, фотоимпульсными и потенциометрическими датчиками манипулятора, а также с двухпозиционными исполнительными механизмами и датчиками оборудования.

Система реализована на базе 16-разрядных микропроцессоров K1801BM1 и обеспечивает функционирование гибкого производственного модуля в автоматическом режиме, режиме обучения и режиме ручного управления. Программное обеспечение системы «Сфера-36» ориентировано на антропоморфный манипулятор РМ-01, имеющий шесть степеней подвижности (три переносных и три ориентирующих). Связь системы с внешней ЭВМ осуществляется по интерфейсу RS232C.

Структурная схема системы «Сфера-36» (рис. 8.5) включает в себя два иерархических уровня. Верхний уровень представляет собой управляющую микроЭВМ со стандартным набором периферийных устройств (видеотерминал, накопитель на гибком магнитном диске, пульт ручного управления) и средств связи с объектом. Нижний уровень выполняет функции управления исполнительными приводами.

Техническая характеристика системы «Сфера-36»

Элементная база	МПК K1801
Разрядность центрального процессора	16
Объем ОЗУ верхнего уровня, К байт	16
Объем ППЗУ верхнего уровня, К байт	32
Объем внешнего ЗУ (НГМД), К байт	65
Разрядность процессора привода	16
Объем ОЗУ нижнего уровня, К байт	2
Объем ППЗУ нижнего уровня, К байт	8
Количество каналов управления приводом	6
Количество каналов дискретного ввода-вывода	40

Аппаратные средства верхнего уровня включают в себя модули центрального процессора (МЦП), ОЗУ, ПЗУ, а также модуль последовательного интерфейса (МПИ), модуль аналогового ввода (МAB), модуль ввода-вывода дискретных сигналов (МВВ) и модуль связи (МС).

Модуль центрального процессора является ведущим по отношению ко всей системе и обеспечивает:

программное управление манипулятором (планирование траектории движения схвата во внешней системе координат, преобразование внешних координат во внутренние, формирование программ движения отдельных звеньев);

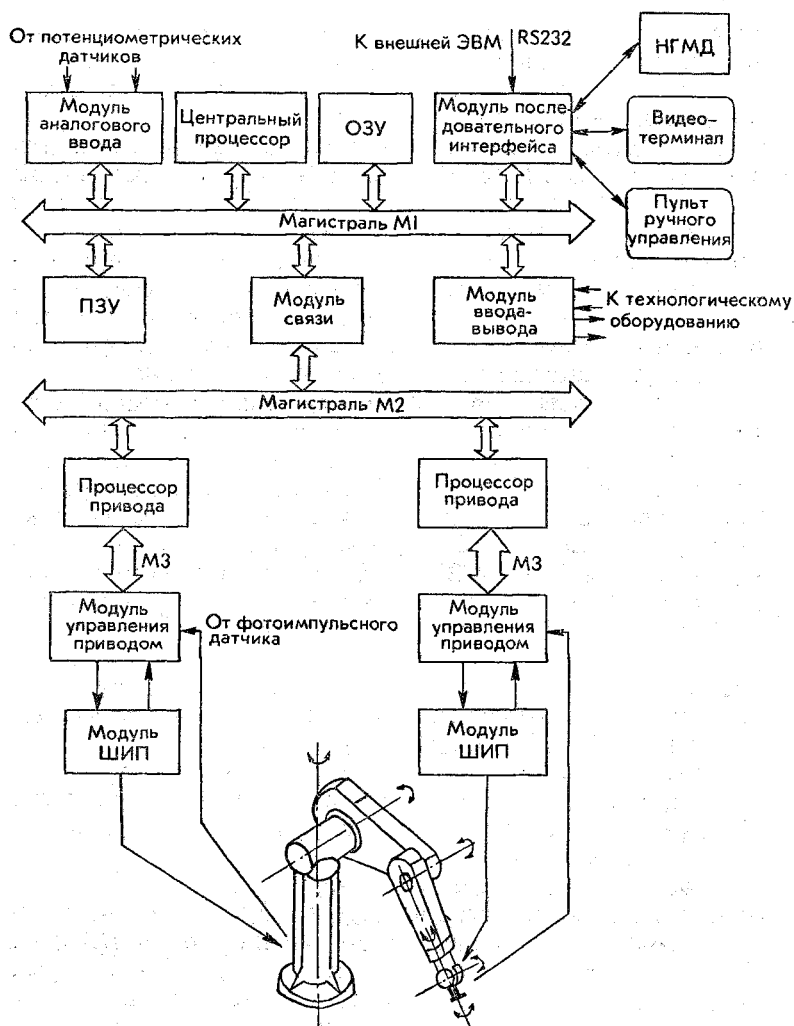


Рис. 8.5

синхронизацию робота с технологическим оборудованием (обработка двухпозиционных сигналов от датчиков оборудования и выдача двухпозиционных команд управления);

организацию диалоговой связи с оператором (в режи-

мах обучения, ручного управления, аналитического программирования);

диагностику и калибровку начального положения звеньев манипулятора.

В модуле центрального процессора размещаются также системное ПЗУ объемом 4 К байт, системное ОЗУ объемом 32 байта и резидентное ПЗУ пользователя объемом 8 К байт. Системное ПЗУ является скрытым (недоступным для программ пользователя) и предназначено для хранения программ начального пуска, обслуживания пультовых режимов, загрузки с перфоленты и гибкого магнитного диска. Системное ОЗУ необходимо для работы системных программ и организации переходов между системной программой и программой пользователя. В резидентном ПЗУ пользователя запоминаются рабочие программы, составленные на языке программирования робота. Для обеспечения режима реального времени в состав центрального процессора входит программируемый таймер, формирующий сигналы прерывания с периодом от 2 до 32 мс для верхнего уровня и с периодом от 0,2 до 3,2 мс для нижнего уровня.

Модуль ПЗУ предназначен для хранения операционной системы и констант алгоритмов управления. Он построен на базе микросхем с ультрафиолетовым стиранием и электрической записью информации. Объем модуля ПЗУ — 32 К байт.

Модуль ОЗУ используется для временного хранения команд, операндов и рабочей программы пользователя. Он выполнен на микросхемах статического типа и имеет объем 16 К байт.

Модуль последовательного интерфейса обеспечивает подключение к магистрали верхнего уровня периферийных устройств, имеющих выход на интерфейс RS232C. Он имеет четыре идентичных канала, три из которых выделены для видеотерминала, накопителя на гибком магнитном диске и пульта ручного управления, а четвертый зарезервирован для связи с внешней ЭВМ. Каждый канал модуля содержит по четыре программно-доступных регистра (регистры состояния приемника, данных приемника, состояния передатчика и данных передатчика). Скорость обмена задается переключками и может изменяться в пределах от 75 до 9600 бод (бит/с).

Модуль аналогового ввода предназначен для преобразования аналоговой информации, поступающей от потенциометрических датчиков грубого отсчета, в цифровой

код. Он содержит два программно-доступных регистра (регистры состояния и данных).

Техническая характеристика МАВ

Количество каналов аналогового ввода	6
Диапазон входных напряжений, В	0—5
Разрешающая способность, бит	8
Время преобразования, мкс	160

Модуль ввода-вывода дискретных сигналов обеспечивает подключение к системе управления двухпозиционных исполнительных механизмов и датчиков, входящих в состав обслуживаемого роботом технологического оборудования. Модуль содержит десять программно-доступных регистров (пять для ввода и пять для вывода), при помощи которых производится запись информации в каналы и считывание их состояния. Параметры сигналов ввода-вывода нормированы с учетом использования стандартных элементов электроавтоматики станков.

Техническая характеристика МВВ

Количество каналов дискретного ввода	40
Количество каналов дискретного вывода	40
Уровни входных сигналов	ТТЛ
Выходное напряжение, В	До 30
Выходной ток, мА	До 40

Обмен данными между верхним и нижним уровнями осуществляется через *модуль связи* в режиме прямого доступа в память процессора привода. Причем инициатором обмена может быть как центральный процессор, так и процессор привода, выдающий в МС запрос на прерывание. Для организации такого обмена модуль связи имеет три программно-доступных регистра (регистр состояния, регистр номера страницы и регистр прерываний), а в адресном пространстве центрального процессора выделяется одна страница виртуальной памяти (256 слов), которую можно совместить с любой страницей памяти любого периферийного процессора. Совмещение осуществляется путем записи соответствующих кодов в регистр состояния и регистр номера страницы МС.

Аппаратные средства нижнего уровня образованы модулями процессоров приводов (МПП), управления приводами (МУП) и широтно-импульсного преобразования (ШИП), которые обеспечивают реализацию программных движений, рассчитанных на верхнем уровне.

Модуль процессора привода предназначен для обработки цифровой информации в контурах управления электроприводом звеньев. Модуль построен на основе 16-разрядного микропроцессора и содержит дополнительно резидентное ОЗУ объемом 2 К байт и резидентное ППЗУ объемом 8 К байт. ОЗУ используется в качестве стека и для временного хранения информации, ППЗУ — для длительного хранения управляющих программ.

Модуль управления приводом осуществляет преобразование управляющих кодов, поступающих от процессора привода, в последовательность широтно-модулированных импульсов, а также первичную обработку информации от фотоэлектрического (инкрементального) датчика. МУП содержит четыре программно-доступных регистра: регистр состояния, регистр управления, регистр приращения позиции и регистр интервала.

Обработка сигналов от фотоэлектрического датчика заключается в определении величины приращения позиции и интервала времени между двумя позиционными импульсами. В дальнейшем эта информация используется процессором привода для расчета абсолютного значения выходной координаты и скорости ее изменения. Кроме того, по завершении каждого полного оборота фотоэлектрический датчик формирует специальный индексный импульс, который подается на линию прерывания процессора привода и используется при калибровке (определении начального положения) манипулятора и контроле исправности датчика.

Модуль широтно-импульсного преобразования представляет собой силовой мост, в диагональ которого включен исполнительный электродвигатель. Цепь питания моста содержит резистивный датчик, используемый для управления схемой токоограничения МУП.

Техническая характеристика модуля ШИП

Выходное напряжение, В	—35—+35
Выходной ток, А	До 5
Частота ШИМ, кГц	8
Дискретность ШИМ, бит	8

Перечисленные модули связываются друг с другом при помощи системных магистралей М1, М2, М3. Магистраль М1 предназначена для обмена информацией между центральным процессором, памятью и внешними устрой-

ствами, магистраль М2 — между модулем связи и процессорами привода, магистраль М3 — между процессорами привода и модулями управления приводом. Состав линий магистралей и временные диаграммы сигналов соответствуют в основном стандарту магистрали МПИ (Q-bus, канал обмена микроЭВМ «Электроника-60»).

Система «Сфера-36» рассчитана на управление манипулятором РМ-01, работающим в угловой (ангулярной) системе координат. Манипулятор имеет шесть степеней подвижности, обеспечивающих позиционирование объекта в заданной точке рабочего пространства и его ориентацию по отношению к базовой системе координат. Приводы манипулятора выполнены на базе электродвигателей постоянного тока с возбуждением от постоянных магнитов и размещаются непосредственно в звеньях. Для переносных степеней подвижности используются электродвигатели мощностью 80 Вт, для ориентирующих 30 Вт. Передача движения от двигателей к звеньям осуществляется через понижающие редукторы. Фотоэлектрические датчики располагаются на валах электродвигателей, потенциометрические — на выходных валах редукторов. Разрешающая способность инкрементальных датчиков — около 200 имп./оборот, потенциометрических — 8 двоичных разрядов.

Для манипулятора РМ-01 разработана система программирования ARPS, реализуемая аппаратными средствами верхнего уровня. Система ARPS является резидентной и позволяет описать движение робота путем задания последовательности положений его звеньев либо координат схвата. Программирование в системе ARPS может производиться как аналитически, так и методом обучения.

Система ARPS состоит из двух основных компонентов — командного языка и языка программирования. Кроме того, программисту доступна особая группа команд для выполнения специальных операций и редактирования, а также команды пультного терминала, позволяющие взаимодействовать с системой на уровне машинных кодов.

Командный язык используется для управления режимами работы системы, запуска и останова программ, выполнения операций с накопителем на гибком магнитном диске и других функций. Команды вводятся с клавиатуры в ответ на сообщение монитора «>» или «RUN>», первое из которых соответствует неподвижному манипуля-

тору, второе — выполнению управляющей программы, в процессе которого манипулятор совершает предписанные движения. При вводе команда должна быть представлена в следующем формате:

имя команды аргумент 1, аргумент 2, ..., где имя может усекаяться до одного, двух или трех символов, а аргументами являются переменные, определяющие положение и ориентацию схвата, имя файла и т. д. Основные команды монитора ARPS приведены ниже:

CHANGE	— изменить значение переменной, определяющей положение схвата;
HERE	— присвоить переменной текущее значение положения схвата;
WHERE < # >	— вывести на терминал текущее значение положения или обобщенных координат (#);
LTEACH	— присвоить последовательности переменных значения положений схвата, определяемые оператором в режиме обучения;
STORE	— записать текст программы и координаты узловых точек на гибкий магнитный диск;
LOAD	— загрузить в память текст программы и координаты узловых точек, записанные на гибком магнитном диске;
PLIST	— вывести текст программы на экран видеотерминала;
LLIST	— вывести на экран видеотерминала координаты узловых точек траекторий;
LDEL —	— удалить из памяти информацию о координатах точек траектории;
FDEL	— удалить файл с гибкого магнитного диска;
FDIR	— вывести на экран видеотерминала имена всех файлов, хранящихся на гибком магнитном диске;
PDIR	— вывести на экран видеотерминала имена всех программ, находящихся в памяти;
SPEED	— определить скорость движения схвата в абсолютных единицах (мм/с);
SPEED%	— определить масштабный коэффициент, используемый при вычислении действительного значения скорости (%);
CAL	— произвести калибровку манипулятора;
EDIT	— выполнить редактирование программы;
RUN	— начать выполнение управляющей программы;
ABORT	— прервать выполнение управляющей программы;
CONTINUE	— продолжить выполнение прерванной программы;

При включении системы управления на экране видеотерминала появляется текст, заканчивающийся вопросом о режиме запуска (с очисткой ОЗУ или сохранением его содержимого). После окончания начального диалога и завершения операций с памятью на экране терминала

выводится символ «>», означающий готовность системы принимать команды монитора. Далее требуется произвести калибровку манипулятора, в процессе которой на основе показаний датчиков грубого отсчета (потенциометрических) и точного отсчета (фотоэлектрических) определяются начальные значения угловых координат. Калибровка инициируется командой монитора CAL и сопровождается небольшим перемещением схвата.

Управляющая программа записывается на языке программирования, операторы которого имеют следующую структуру:

имя оператора аргумент 1, аргумент 2, ...,
причем имя может состоять из одного либо двух слов, усекаемых до одного, двух или трех символов, а аргумент представляет собой переменную или константу, описывающую положение, расстояние, скорость и другие параметры.

Язык программирования ARPS допускает использование следующих типов переменных и констант: целые, действительные, переменные положения. Для обозначения переменных используются буквы A — Z, цифры 0—9 и знак точки «.». Длина имени не ограничена, но не рекомендуется использовать имя длиной свыше 10 символов. Первым символом имени обязательно должна быть буква.

Целые числа используются при обозначении номеров каналов ввода-вывода, количеств повторений цикла, номеров звеньев и т. д. Они могут изменяться в пределах от —32768 до +32767. Над целыми числами можно производить арифметические операции.

Действительные числа используются при определении расстояний, угловых величин, скорости, времени задержки. Расстояния задаются в миллиметрах и могут лежать в пределах от —1024,00 мм до +1023,99 мм (дискретность 0,01 мм). Угловые величины задаются в градусах и изменяются от —180,000° до +179,995° (дискретность 0,005°). Скорость устанавливается путем задания абсолютных значений (в мм/с) и коэффициента масштабирования (в %) и изменяется в пределах от 2,0 до 3000,0 мм/с (дискретность 0,1 мм/с). Однако реально для манипулятора РМ-01 скорость движения схвата не может превысить 500 мм/с. Время задержки задается в секундах и может изменяться от 0,00 до 327,67 с (дискретность 0,01 с).

Переменные положения определяют расположение

рабочего органа робота во внешней (декартовой) либо внутренней (угловой) системе координат. В первом случае переменная положения состоит из трех декартовых координат x, y, z и трех углов ориентации α, β, γ . Во втором случае эта переменная включает в себя шесть угловых координат joint1, ... joint6, описывающих взаимное расположение соседних звеньев. Чтобы отличать такие переменные при записи во внутренней системе координат перед именем ставится символ #.

Ниже приведены основные операторы программирования языка ARPS:

GO	— переместить схват манипулятора в требуемое положение по траектории, построенной во внутренней системе координат;
GOS	— переместить схват манипулятора в требуемое положение по прямолинейной траектории;
GO NEAR	— переместить схват манипулятора в окрестность заданной точки по траектории, построенной во внутренней системе координат;
GOS NEAR	— переместить схват манипулятора в окрестность заданной точки по прямолинейной траектории;
MOVE	— переместить схват манипулятора на заданное расстояние по траектории, построенной во внутренней системе координат;
MOVES	— переместить схват манипулятора на заданное расстояние по прямолинейной траектории;
OPEN	— открыть схват;
CLOSE	— закрыть схват;
DELAY	— задержать выполнение программы на заданное время;
WAIT IN	— задержать выполнение программы до появления заданной комбинации сигналов на входных линиях;
SHIFT	— переместить начало мировой системы координат;
FRAME	— сформировать новую систему координат, заданную точками рабочего пространства;
SCALE	— выполнить операцию масштабирования координат x, y, z ;
TOOL	— ввести коррекцию на размеры рабочего инструмента;
OUT	— установить в заданное состояние значение сигнала на одной выходной линии;
OUT GROUP	— установить в заданное состояние значение сигналов на 16 выходных линиях;
CALL	— вызвать подпрограмму;
IN CALL	— вызвать подпрограмму в случае сигнала прерывания по определенному каналу;
RETURN	— возвратиться из подпрограммы;
IF	— оператор условного перехода;
JUMP	— оператор безусловного перехода;
SET	— выполнить арифметическую операцию над целыми переменными;

- DISTANCE — вычислить расстояние между двумя точками рабочего пространства;
- PRINT — вывести на экран видеотерминала текст и (или) значение переменной;
- STOP — остановить выполнение программы.

Создание управляющей программы происходит при помощи редактора. Вызов редактора осуществляется командой монитора EDIT, за которой следует выбранное пользователем имя программы. Редактор позволяет также вносить изменения в ранее созданные программы.

В системе ARPS используется редактор с автонумерацией строк, имеющий следующие команды:

- I — вставка строки;
- P — вывод строки на экран;
- D — удаление строки;
- R — замена текста в строке;
- RA — замена текста во всех строках;
- TGO — формирование последовательности команд типа GO, определяющих движение в режиме обучения;
- TGOS — формирование последовательности команд типа GOS, определяющих прямолинейное движение в режиме обучения;
- E — выход из редактора.

Программе, созданной редактором, присваивается имя с расширением «.P». Кроме текста программы требуется определить координаты узловых точек траектории, которые хранятся в файле с тем же именем и расширением «.L». Узловые точки траектории могут быть определены в режиме обучения (команды монитора LTEACH, HERE) либо путем прямого задания координат (команда монитора CHANGE). Возможно также совмещение режимов обучения и редактирования (команды редактора TGO и TGOS).

Длина программы и количество узловых точек траектории ограничены объемом памяти, используемой системой ARPS. В среднем для записи одного оператора требуется 12 байт, для записи точки во внешней системе координат — 36 байт, для записи точки во внутренней системе координат — 22 байта памяти. В ОЗУ верхнего уровня собственно для системы ARPS выделено 4 К байт, а в оставшейся части можно записать программу длиной до 560 шагов и около 500 точек траектории. На гибком магнитном диске размещается 256 блоков по 256 байт в каждом, что соответствует 1000 операторам и 1000 точкам траектории.

Перед выполнением программы требуется задать скорость движения схвата (команды монитора SPEED, SPEED%). Запуск программы осуществляется командой монитора RUN.

В качестве примера рассмотрим простейшую программу, описывающую перемещение объекта из началь-

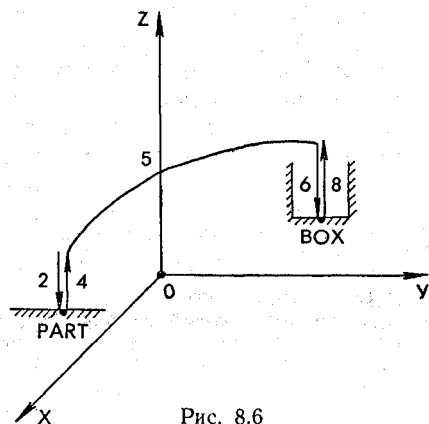


Рис. 8.6

ного положения PART в конечное BOX (рис. 8.6). Текст программы имеет следующий вид:

1. GONEAR PART, 50
2. GOS PART
3. CLOSE
4. GOSNEAR, 50
5. GONEAR BOX, 100
6. GOS BOX
7. OPEN
8. GOSNEAR, 100

В соответствии с приведенной программой задача перемещения объекта распадается на ряд более мелких операций.

1. Схват робота помещается над деталью, на 50 мм выше ее.
2. Схват перемещается вплотную к детали.
3. Осуществляется захват детали.
4. Деталь приподнимается на 50 мм.
5. Производится перемещение детали в позицию, расположенную на 100 мм выше заданной.
6. Деталь опускается в заданную позицию.
7. Осуществляется разжатие схвата.
8. Схват приподнимается на 100 мм над деталью.

Приведенную программу можно повторить неограниченное число раз, при условии, что обеспечивается автоматическая подача новой детали в позицию PART и автоматическое извлечение детали из позиции BOX.

8.4. Устройство числового программного управления на базе микроЭВМ «Электроника-60»

Появление микропроцессоров и микроЭВМ привело к качественным изменениям в технике управления металлорежущими станками. Впервые специалисты получили возможность унифицировать аппаратные средства УЧПУ для различных групп станков, сократить время их проектирования и изготовления. Группа УЧПУ, созданных на базе микроЭВМ «Электроника-60», включает в себя устройства 2M43, 2C85, 2C42, 2P22, 2P32 и другие, используемые для автоматического управления как простейшими, так и сложными многооперационными станками.

Наиболее совершенной моделью УЧПУ на базе микроЭВМ «Электроника-60» является устройство 2C42, предназначенное для управления обрабатывающими центрами и координатно-расточными станками. Устройство рассчитано на подключение следящего электропривода подач постоянного тока с индуктивными либо фотоэлектрическими измерительными преобразователями и встроенными датчиками скорости. Устройство 2C42 обеспечивает контурную обработку детали одновременно по трем координатам и позволяет вводить коррекцию на радиус и длину инструмента, изменять скорости главного движения и рабочих подач, осуществлять редактирование программы, выдачу информации на устройство отображения, выдачу технологических команд. Устройство может производить тестовый контроль функциональных узлов и адаптивное управление по двум каналам. Для связи устройства 2C42 с внешней ЭВМ используется интерфейс ИРПС.

Техническая характеристика УЧПУ 2C42

Элементная база	МПК К581
Разрядность микропроцессора	16
Объем ОЗУ, К байт	10
Объем ПЗУ, К байт	24
Количество управляемых координат	8
Количество каналов дискретного ввода-вывода	До 160
Количество каналов аналогового ввода	2
Программноноситель	8-дорожечная перфолента

Структурная схема устройства 2С42 представлена на рис. 8.7. Основным узлом устройства является микроЭВМ «Электроника-60», включающая в себя центральный процессор, ПЗУ, ОЗУ и интерфейсные модули. Процессор выполняет все необходимые арифметико-логические операции по управлению станком и распределению

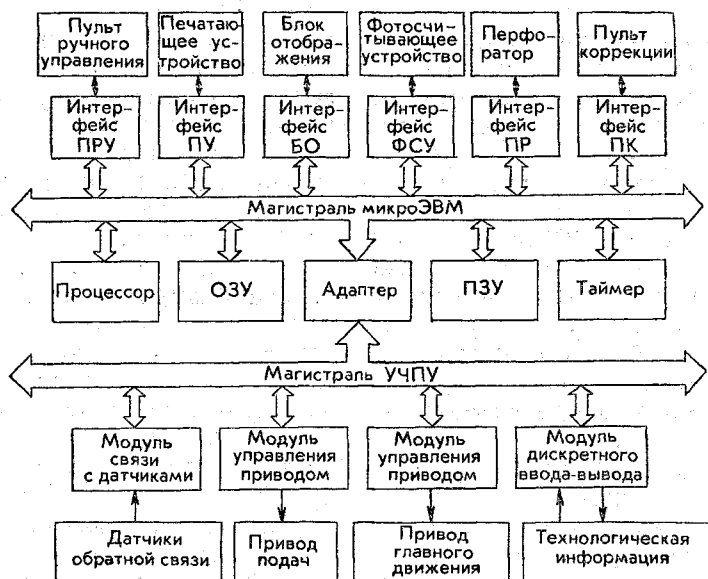


Рис. 8.7

канала между внешними устройствами и блоками. ПЗУ служит для хранения технологических и функциональных программ, ОЗУ — для организации вычислительного процесса и записи промежуточных результатов. Сохранение информации в ОЗУ при отключении основного питания достигается за счет использования резервных аккумуляторов.

Центральный процессор построен на микропроцессорном наборе К581, состоящем из БИС регистрового АЛУ, БИС управления и БИС постоянного запоминающего устройства микрокоманд. Элементы набора связаны друг с другом 22-разрядной шиной. Синхронизация работы всех БИС обеспечивается четырьмя последовательностями тактирующих сигналов. Центральный про-

цессор выполняет одно- и двухадресные команды и может обрабатывать как 16-разрядные слова, так и 8-разрядные байты.

К микроЭВМ «Электроника-60» через соответствующие интерфейсы подключаются пульт коррекции, пульт ручного управления, устройство печати, блок отображения символьной информации, фотосчитывающее устройство и перфоратор. Пульт коррекции предназначен для изменения скорости главного движения и скорости подачи соответственно в пределах от 40 до 140 % и от 0 до 120 % от запрограммированных величин. Пульт ручного управления позволяет задавать режим работы устройства, производить ручной ввод данных, осуществлять редактирование программы. Блок отображения символьной информации (6 строк по 32 знака в строке) выполнен на базе электронно-лучевой трубки и используется при вводе программы и ее редактировании. Устройство печати обеспечивает получение твердой копии программы. Фотосчитывающее устройство предназначено для считывания управляющей программы с перфоленты, перфоратор — для вывода откорректированных вариантов программы и получения дубликатов перфолент.

Для обеспечения режима реального времени в устройстве используется программируемый таймер с частотой задающего генератора 100 кГц, позволяющий формировать периодические сигналы прерывания и производить отсчет временных интервалов с длительностью от 0,7 с до 10 с.

Электрооборудование станка подключается к микроЭВМ через интерфейсные модули, имеющие выход на отдельную магистраль (см. рис. 8.7). Магистрали УЧПУ и микроЭВМ соединяются друг с другом через специальный адаптер. Блоки управления приводами формируют выходные сигналы в виде напряжения постоянного тока с диапазоном от -10 до $+10$ В. При формировании сигнала управления приводом используется информация об ошибке по положению, скорости, а в случае адаптивного управления — и о значении крутящего момента. Модуль связи с датчиками перемещения осуществляет преобразование их выходных сигналов в цифровой код. Блоки дискретного ввода-вывода обеспечивают бесконтактную выдачу двухпозиционных сигналов на электроавтоматику станка, прием сигналов от станка, а также обмен информацией с другим технологическим оборудованием.

Используемая в устройстве микроЭВМ в совокупности с соответствующим математическим обеспечением и интерфейсными модулями реализует вычисление траектории и скоростей перемещения рабочих органов станка, выдачу управляющих воздействий, контроль за их работой и другие функции. Управляющая информация вводится с восьмидорожечной перфоленты либо вручную с пульта управления. Информация кодируется в стандарте ISO. Задание размеров может осуществляться в абсолютных значениях и приращениях. При построении траектории может использоваться линейная либо круговая интерполяция.

Устройства ЧПУ на базе микроЭВМ «Электроника-60» объединены в серии 2С, 2Р, 2М и выпускаются как в стойном, так и блочно-модульном исполнении, с выносным дисплейным блоком, устанавливаемым непосредственно на станке. Модификации устройств этих серий отличаются в основном числом управляемых координат и составом технологического программного обеспечения, учитывающего специфику обработки на отдельных группах станков. В сериях 2С, 2Р, 2М выпускаются устройства ЧПУ для управления как следящим, так и шаговым приводом.

8.5. Устройство числового программного управления «Электроника НЦ-31»

УЧПУ «Электроника НЦ-31» предназначено для управления токарными станками со следящим приводом подачи и импульсными датчиками обратной связи. Оно обеспечивает контурное управление по двум координатам с линейной и линейно-круговой интерполяцией при задании размеров в приращениях и абсолютных значениях. Устройство оснащено цифровой индикацией и позволяет осуществлять коррекцию программы, смещение нуля станка, ввод информации в метрической и дюймовой системе измерений. Конструктивно устройство «Электроника НЦ-31» выполнено в виде объединенного блока, который встраивается в специальный отсек станка.

Техническая характеристика УЧПУ «Электроника НЦ-31»

Элементная база	МПК К588
Разрядность микропроцессора	16
Объем ОЗУ, К байт	8
Объем ПЗУ, К байт	16
Объем кассеты внешней памяти, кВ	8
Число управляемых координат	2
Число каналов дискретного ввода-вывода	32

В состав устройства (рис. 8.8) входят два модуля процессоров со встроенным ПЗУ, модуль ОЗУ, пульт оператора, модуль адаптера каналов и таймера, контроллеры импульсных преобразователей, электроавтоматики и приводов. Все модули соединены между собой магистралью МНЦ, построенной по принципу «общей шины».

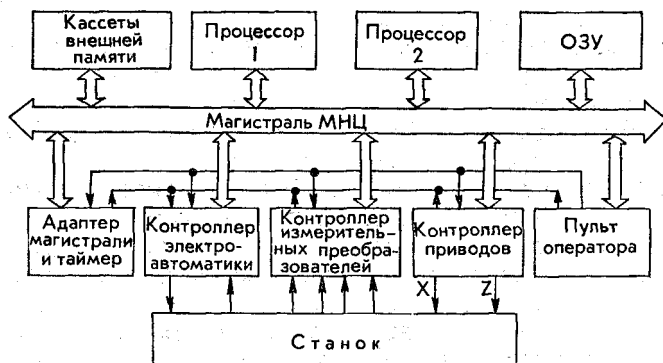


Рис. 8.8

Кроме того, имеется дополнительный радиальный канал, управляемый адаптером магистрали. К дополнительному каналу подключаются пульт управления и модули контроллеров. Такая архитектура (с дополнительным каналом) увеличивает функциональную гибкость устройства и упрощает аппаратную реализацию модулей.

Процессоры 1, 2 работают совместно и выполняют все операции по расчету траектории движения инструмента и управлению следящим приводом подач. Каждый из процессоров содержит в своем составе резидентное ПЗУ, используемое для хранения системных программ. Обмен информацией между процессорами производится через общее ОЗУ.

Адаптер каналов и таймер выполняют функции преобразования магистрали в радиальный канал и отсчета программно задаваемых интервалов времени. Задание на отработку временных интервалов может поступить от любого из процессоров, по окончании отработки формируется сигнал прерывания соответствующего процессора. Временной интервал задается с дискретностью 0,1 мс; максимально возможная длительность интервала — 64 К дискрет.

Контроллер электроавтоматики осуществляет переключение исполнительных реле станка и прием сигналов состояния с контактов реле и кнопок управления. Модуль связан со станком 16 входными и 16 выходными линиями. Выходные линии рассчитаны на коммутацию тока до 0,2 А при напряжении 27 В. Восемь входных линий можно использовать для организации прерываний. Все входные и выходные линии гальванически отделены (через оптроны) от электрических цепей станка.

Контроллер импульсных преобразователей предназначен для приема сигналов от фотоэлектрических импульсных датчиков, расположенных на приводах, шпинделе, штурвале, и преобразования их в параллельный двоичный код. Показания этих датчиков используются при определении линейных перемещений режущего инструмента по осям x , z , частоты вращения детали и относительного перемещения резца, задаваемого оператором вручную.

Контроллер привода обеспечивает управление скоростью движения (подачи) режущего инструмента по осям x , z , выполняя при этом функцию преобразования 12-разрядного кода скорости в пропорциональный аналоговый сигнал с диапазоном от -10 до $+10$ В.

Пульт оператора содержит ряд органов управления и индикаторов, служащих для ввода, редактирования, отладки и исполнения программы, управления режимами работы станка, а также отображения информации о скорости подачи, номере кадра программы, параметров состояния ЧПУ.

УЧПУ «Электроника НЦ-31» имеет специальное программное обеспечение, ориентированное на выполнение функций токарной обработки. Оно включает в себя ряд технологических циклов, оформленных в виде G-функций. К ним относятся циклы продольного и поперечного точения, глубокого сверления, нарезания торцевых и цилиндрических канавок, нарезания резьбы. Предусмотрена также возможность параметрического задания подпрограмм циклов и выполнения команд условного перехода по внешнему сигналу.

Управляющая программа может вводиться непосредственно с клавиатуры пульта управления либо переносится с другого станка или устройства подготовки программ при помощи кассеты внешней памяти, представляющей собой модуль ОЗУ с автономным источником питания. Объем кассеты внешней памяти — 8 К байт, время сохранения информации — не менее 100 ч.

Программное обеспечение УЧПУ «Электроника НЦ-31» позволяет также осуществлять подготовку управляющей программы в режиме обучения. При этом параллельно с обработкой детали, выполняемой при ручном управлении станком, автоматически формируется управляющая программа и производится ее разбиение на кадры.

Дальнейшее совершенствование устройств ЧПУ привело к появлению мультипроцессорных устройств блочно-модульного типа, обладающих повышенной гибкостью и реализующих ряд новых функций. Примером таких устройств является устройство числового программного управления «Электроника НЦ80-31» («Электроника МС2101»), которое может использоваться для управления токарными, шлифовальными станками, фрезерно-сверлильными обрабатывающими центрами и гибкими производственными модулями. В зависимости от типа станка и задач управления в состав УЧПУ могут входить два или три блока, кроме того, можно варьировать составом плат некоторых блоков.

Первый блок (дисплейный) во всех исполнениях имеет одинаковую аппаратную часть и отличается только программным обеспечением. Он включает вычислитель, реализованный на микропроцессорном комплекте К1801, модуль последовательного интерфейса и запоминающее устройство на цилиндрических магнитных доменах. К первому блоку подключается клавиатура и плазменный дисплей.

Второй блок предназначен для управления приводами, электроавтоматикой станка и хранения технологического программного обеспечения. В его состав входят вычислитель, контроллеры электроавтоматики и приводов, аналого-цифровой преобразователь. Дисплейный блок связывается со вторым блоком телеграфным каналом, интерфейс которого размещается на плате вычислителя. Скорость обмена информацией между блоками — 19,2 К бод.

Программное обеспечение УЧПУ «Электроника НЦ80-31» содержит интерпретатор языка программирования технологических функций ПЛ-ЧПУ, компилятор языка программирования электроавтоматики «ЯРУС-2», средства ввода, редактирования, отладки и исполнения управляющих программ, а также диагностические программы.

8.6. Микропроцессорная система управления АСУ-ВИ

Концепция построения. Автоматизированные системы контроля и испытаний (АСКИ) на базе типовых средств вычислительной техники не дают возможности пользователям самостоятельно освоить, эксплуатировать и развивать данные системы без привлечения профессиональ-

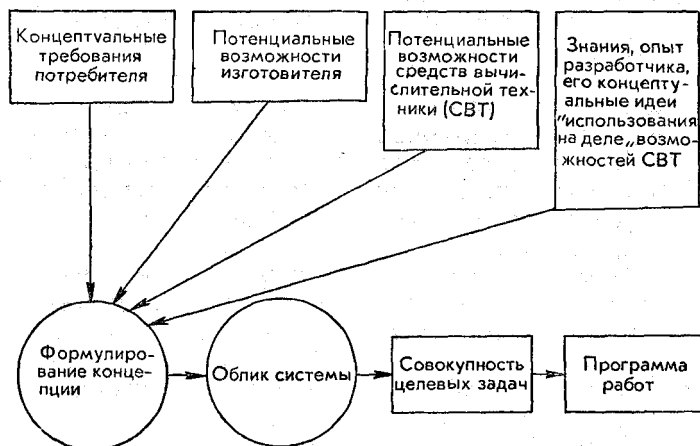


Рис. 8.9

ных специалистов. В то же время специалисты по АСКИ не могут в короткие сроки и в массовом порядке создавать достаточно надежные «дружественные» системы. Поэтому необходимо создать такую методологию и инструментальные средства АСКИ, которые позволили бы не только делать системы рассматриваемого класса лучше и быстрее, но и давали бы количественные оценки их качества, позволяющие судить, насколько они эффективны. Одно из возможных решений указанной проблемы сформулировано в так называемой *концепции ПИР-требований*. Ее сущность заключается в следующем: развитие АСКИ должно осуществляться так, чтобы удовлетворять росту соподчиненных требований пользователя (П), изготовителя (И) и разработчика (Р) (рис. 8.9).

Основные требования к АСКИ: 1) АСКИ не должна мешать пользователю; 2) АСКИ должна помогать пользователю; 3) АСКИ должна гарантировать достовер-

ность результатов; 4) АСКИ должна быть конкурентоспособна; 5) АСКИ должна быть серийноспособна; 6) АСКИ должна быть проста в наладке; 7) АСКИ должна быть обеспечена стендовым и технологическим оборудованием; 8) АСКИ должна быть не ниже уровня мировых образцов; 9) АСКИ должна развиваться; 10) процесс проектирования новых версий АСКИ должен экономить средства и время разработчика.

Общая потребность в АСКИ только по важнейшим отраслям в СССР огромна. Вместе с тем каждая АСКИ должна учитывать специфику предприятия, т. е. должна быть *индивидуальна*. Это требование пользователя вступает в противоречие с требованиями изготовителя, так как снижает серийноспособность выпускаемых АСКИ и затрудняет обеспечение единства методов и средств измерения при их аттестации. Кроме того, для индивидуального проектирования необходимо большое количество высококвалифицированных разработчиков систем указанного назначения.

Указанное противоречие частично устраняется при *групповом* проектировании, в результате которого создается типовый проект для группы родственных предприятий (производств). Как правило, такой проект оказывается «эффективным в среднем по группе».

В основе современного *массового* проектирования лежит создание унифицированных средств для комплектования АСКИ по спецификации заказчика. В этом случае развитие АСКИ идет по двум направлениям: приборному (с ЭВМ и без нее) и функционально-узловому.

Первое направление отражает результат эволюции виброизмерительной техники по линии «автономный прибор — прибор системного применения — автоматизированная система контроля и испытаний». Оно удовлетворяет растущие потребности в автономных цифровых электроизмерительных приборах и обеспечивает требуемый уровень автоматизации (при организации АСКИ по принципам измерительно-вычислительного комплекса). Однако по мере усложнения объектов исследования все ощутимее проявляются недостатки первого направления: трудности создания недорогих и компактных многоканальных систем; сложность сопряжения приборов различных фирм, использующих собственные стандарты на электрическую и конструктивную совместимость; низкий коэффициент использования изме-

рительных приборов при постоянно сохраняющемся их дефиците и пр.

Второе направление — результат развития АСКИ по линии «персональные ЭВМ (ПЭВМ) — профессиональные персональные ЭВМ (ППЭВМ) — автоматизированные системы контроля и испытаний». Оно обеспечивает повышение ряда показателей качества АСКИ (табл. 8.1). Однако развитие этого направления сдерживается необходимостью проведения большого комплекса работ по унификации и метрологической аттестации алгоритми-

Таблица 8.1

Характеристика АСКИ	Принцип организации			
	приборный		функционально-узловой	
	с приборным интерфейсом	с ЭВМ и ее системным интерфейсом	программная специализация	аппаратурно-программная специализация
Уровень основных функциональных характеристик	1*	1	2	2
Уровень автоматизации	2	1	1	1
Уровень сервиса	2	1	1	1
Стоимость	3	4	1	2
Требуемый уровень квалификации пользователя	1	3	2	2
Сложность технического обеспечения (ТО)	3	4	1	2
Сложность программного обеспечения	1	2	3	2
Способность к самоконтролю	4	3	1	2
Коэффициент использования ТО	2	2	1	1
Способность к переналадке	2	1	1	1
Способность к развитию	3	2	1	2
Живучесть	1	1	3	2
Габариты и масса	3	4	1	2
Потребляемая мощность	3	4	1	2
Операционность	4	3	1	2
Интегральная оценка	35	36	21	25
Средний показатель	2,33	2,4	1,4	1,67

* Уровни качества определяются путем сравнения АСКИ, построенных на основе перечисленных принципов. Наилучший уровень качества — 1, наихудший — 4.

ческих и программных модулей, схемотехнических и конструкторско-технологических решений, разработке соответствующих методических и нормативно-справочных документов. Отсутствие единых методов измерения параметров специфицированных комплексов и многообразие используемой при этом измерительной аппаратуры ведут к различию и невозможности сопоставления результатов аттестации таких АСКИ.

Анализ направлений «массового проектирования» АСКИ (табл. 8.1) показывает, что «оптимальным в среднем» является программная специализация ПЭВМ. Однако, учитывая, что при этом не обеспечивается требуемый уровень основных функциональных характеристик, решение следует искать в области аппаратно-программной специализации. Для сокращения числа вариантных решений перспективным представляется создание типовых проектов АСКИ с последующей разработкой на их основе индивидуальных проектов или «привязкой» типовых проектных решений к условиям конкретного производства. Такой подход до настоящего времени не давал эффекта из-за «жесткости» этих решений, т. е. неуниверсальности, невозможности обеспечить с их помощью без существенных доработок (эквивалентных по трудоемкости индивидуальному проектированию) функционирование АСКИ в условиях, отличающихся от заданных ранее при проектировании, либо существенной сложности универсальных систем, ориентированных на программную реализацию алгоритмов АСКИ и резкое повышение требований к компьютерной грамотности испытателя. Отрицательно сказывались на попытках широкого внедрения АСКИ также низкая автоматизация отдельных производств, а при наличии АСУП — отсутствие решений по единой информационной базе, ограниченные возможности алгоритмического, программного и технического обеспечения и др.

Новый подход к типизации заключается в создании такой системы (называемой системой развития — СР), которая посредством усечения по всем (либо любым) ее обеспечениям: техническому, программному, лингвистическому, метрологическому, — может быть адаптирована с условием конкретного производства. По требованию пользователя можно специализировать системы программным и аппаратно-программным путем (см. табл. 8.1).

Программная специализация предполагает создание

ППП, обеспечивающего решение задачи на базе ядра СР (серийно выпускаемой ПЭВМ).

В основе аппаратно-программной специализации лежит распределение функций и вычислительных мощностей между ядром СР и ее «оболочкой» — специализированными аппаратными модулями (САМ).

При таком подходе практически все ПИР-требования удовлетворяются разработчиком в процессе создания специализированных технического и программного обеспечения АСКИ.

Малые аппаратные затраты и высокие технические характеристики САМ достигаются при использовании принципов специализации пары преобразований и совмещения подсистем.

Специализация пары преобразований «синтез-анализ» по видам испытательных воздействий. Пусть $z(1), z(2), \dots, z(k), \dots, z(K)$ — возможные архитектурные решения АСКИ. Каждая архитектура $z(k)$ описывается множеством структурных модулей M_k и графом S_k их связей, т. е. $z(k) = \{M_k, S_k\}$, и обеспечивает достижение технических характеристик $C_k = \{C_{ijk}\}$, $i = 1, 2, \dots, I$; $j = 1, 2, \dots, J_i$, где I — количество видов испытаний, реализуемых АСКИ; J_i — число критериев оценки АСКИ, ориентированной на проведение i -го вида испытаний. Эффективность достижения C_{ijk} в $z(k)$ будем оценивать с помощью E -критерия (для упрощения записи полагаем, что развитие АСКИ идет по пути увеличения C_{ijk}):

$$E_{ijr} = \delta_{ijk} E_{ij} + \beta_{ij} \text{Sign}(C_{ijk} - C_{ijk}),$$

где δ_{ijk} — относительное отклонение достигнутого значения i -й технической характеристики C_{ijk} в $z(k)$ от требуемой величины C_{ijk} , $\delta_{ijk} = (C_{ijk} - \bar{C}_{ijk}) / \bar{C}_{ijk}$; E_{ij} — значение весового коэффициента, учитывающего важность j -й характеристики для i -го вида испытаний; β_{ij} — величина штрафа (безразмерная величина) за недостижение \bar{C}_{ijk} значения C_{ijk} ; $\text{Sign}(x)$ — знаковая функция вида $\text{Sign}(x) = \{0, \text{ для всех } x \leq 0; 1, \text{ для всех } x < 0\}$. Тогда решением задачи построения АСКИ будет нахождение архитектурного решения $z(I)_{\text{opt}} (E_i \in \{z(k)\})$, оптимального по выбранному критерию E_i для проведения I различных (требуемых) видов испытаний, т. е. удовлетворяющего условию

$$\bar{E}_I = \min_{j_1} \sum_{i_1=1}^{J_1} \dots \sum_{i_I=1}^{J_I} E_{ijk}. \quad (8.1)$$

Создание АСКИ без уменьшения величин K и I существенно затруднено. Поэтому на практике поиск $z(I)_{\text{opt}}$ целесообразно осуществлять в пределах принятой концепции построения систем данного класса (ограничение K) и обусловленных специализаций АСКИ (ограничение I). В частности, при создании проблемно-ориентированных систем условие (8.1) преобразуется к виду

$$E_R = \min \sum_{j_i=1}^{J_i} \dots \sum_{j_R=1}^{J_R} E_{rjk}, \quad R < I, \quad (8.2)$$

а при создании специализированных АСКИ — к виду

$$E_i = \min_k \sum_{j=1}^{J_i} E_{ijk}. \quad (8.3)$$

Очевидно, что

$$E_i \leq E_R \leq E_I, \quad (8.4)$$

т. е. чем выше специализация, тем эффективнее по E -критерию средство. Поскольку в реальных условиях эксплуатации АСКИ в каждый конкретный момент времени осуществляется, как правило, один вид испытаний, предпочтительнее оказывается разработка АСКИ, специализированной по видам испытаний, у которой $E_i(R) = \{E_i\}$, $i = 1, 2, \dots, R$.

Реализация модульного принципа, при котором каждому M_l для всех $l \in \{1, 2, \dots, k, \dots, L_k\}$, где L_k — число структурных модулей, ставится в соответствие один или несколько элементов матрицы C_k (т. е. каждый M_l обеспечивает достижение одной или нескольких из требуемых характеристик C_{ij} , позволяет провести группировку C_{ij} по определенному признаку для последующего обеспечения пересекающихся C_{ij} обобщенными по видам испытаний средствами (как правило, широкого применения с априорно определенной архитектурой, т. е. $z(O) = \{M_O, S_O\}$). При этом условия (8.1) и (8.2) могут быть записаны соответственно в виде:

$$\hat{E}_I = E_O + \sum_{i=1}^I E_{Si}; \quad (8.5)$$

$$E_R = E_O + \sum_{i=1}^R E_{Si}, \quad (8.6)$$

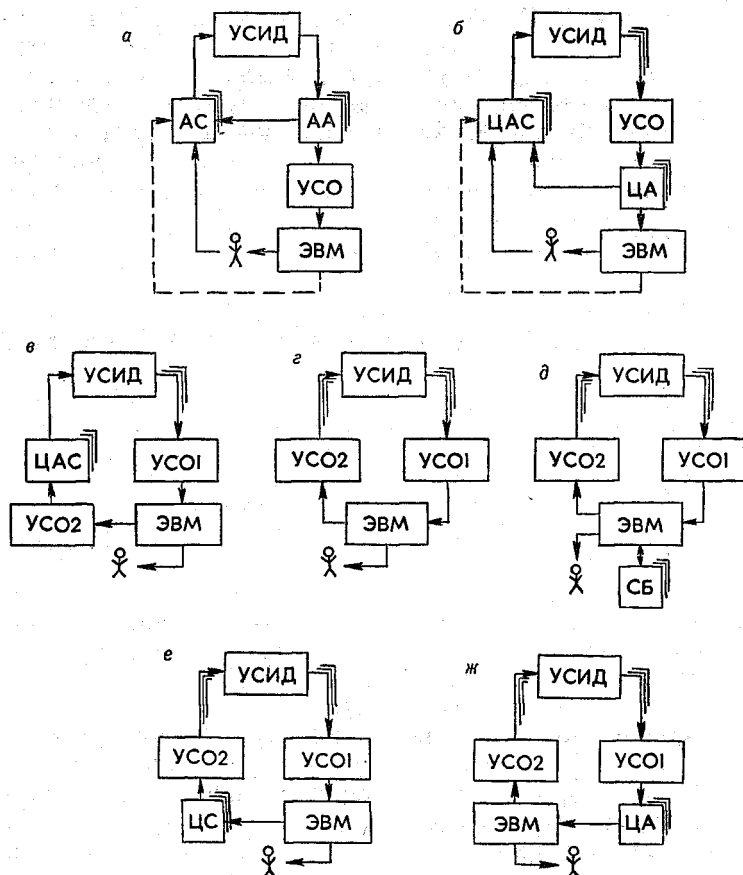


Рис. 8.10

где E_o — E — критерий обобщенных средств; E_{Si} — E — критерий специальных средств, выбирается из соотношения (8.3) при условии, что k определяет лишь возможные архитектурные решения специальных средств $z(k)$, а j — номера элементов C_{ij} , непокрытых $z(O)$. Оптимум характеристик в проблемно-ориентированной АСКИ ($z(R)$) достигается при специализации модулей ($z(R) = z(R) \setminus z(O)$).

Аппаратурная специализация функциональных частей АСКИ известна уже в младших поколениях комбинированных (централизованно-иерархических) систем управления (рис. 8.10, а — в, е, ж). Однако даже наибо-

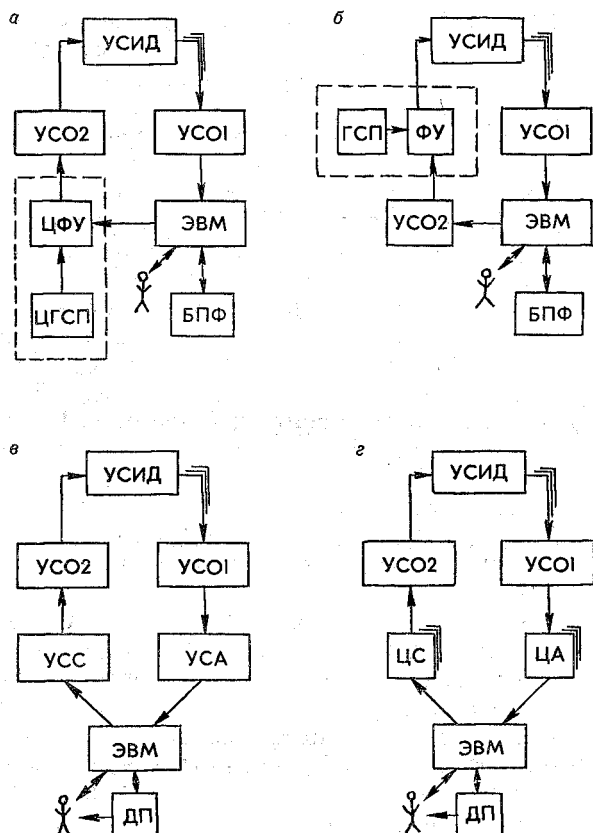


Рис. 8.11

более перспективные из них, содержащие процессор быстрого преобразования Фурье (БПФ), цифровые генератор случайных процессов (ЦГСП) и формирующее устройство (ЦФУ) (рис. 8.11, а) либо аналоговые генератор случайных процессов (ГСП) и формирующее устройство (ФУ) (рис. 8.11, б), накладывают жесткие ограничения на минимально допустимую производительность подсистемы централизованного управления и ее системы ввода-вывода. Реализация иерархической структуры АСКИ (рис. 8.11, в) снижает указанные ограничения за счет ужесточения требований к производительности, точностным характеристикам и функциональным возможностям

УСС и УСА ($\tilde{E}_I = E_O + E_{SI}$) в случае использования универсальных средств анализа (УСА) и синтеза (УСС). Указанное противоречие частично устраняется при создании совокупности процессоров анализа и синтеза, попарно ориентированных на определенный вид испытаний (рис. 8.11, з), например испытания на гармоническое, либо случайное, либо импульсное воздействие. Причем соотношение (8.4) указывает на то, что непересекающиеся C_{ij} целесообразно обеспечивать специальными средствами (модулями M_i), ориентированными на реализацию C_{ij} . Это упрощает точностный анализ генераторов испытательных сигналов, обеспечивает уменьшение погрешности, увеличение динамического диапазона и упрощает алгоритмы формирования испытательных воздействий.

Совмещение подсистем синтеза и анализа. Пусть $A_{h1}, A_{h2}, \dots, A_{hn}$ — записанные на языке h алгоритмы формирования входных воздействий, подаваемых на входы испытываемого устройства. Для каждого A_{hi} имеется отображение $A_{hi} \rightarrow \{A_{hi1}, A_{hi2}, \dots, A_{himi}\}$, где $A_{hi1}, A_{hi2}, \dots, A_{himi}$ — возможные алгоритмы (в языке h) анализа выходных реакций испытываемого устройства при входных воздействиях, описываемых алгоритмом A_{hi} . Сложность аппаратуры, реализующей алгоритм A_h , будем оценивать с помощью критериев k_1, k_2, \dots, k_p , которые отражают, например, число корпусов микросхем, глубину схемы и т. д., причем сложность по критерию $k_l, l \in \{1, 2, \dots, P\}$, аппаратуры, реализующей алгоритм A_h , обозначим $L_{kl}(A_h)$. Известно, что сложность совместной схемной реализации двух алгоритмов, к примеру A_{hi} и A_{hj} , за счет выделения одинаковых операций, реализуемых на общем оборудовании, как правило, меньше суммы сложностей реализации алгоритмов A_{hi} и A_{hj} в отдельности, т. е. $L_{kl}(A_{hi}, A_{hj}) \leq L_{kl}(A_{hi}) + L_{kl}(A_{hj})$. В связи с этим на практике возникает следующая задача. Для каждого $A_{hi}, i = 1, 2, \dots, n$, необходимо выбрать такой алгоритм A_{hik} , при котором совместно удовлетворяется условие $L_{kl}(A_{hi}, A_{hir}) \leq L_{kl}(A_{hi}, A_{hij})$ для всех $j = 1, 2, \dots, m_i$, что обеспечит оптимальную по критерию k_l совместную аппаратурную реализацию каждой пары $(A_{hi} + A_{hj}), i = 1, 2, \dots, n$ (локальный оптимум). Глобальным оптимумом по критерию k_l будет обладать то решение, которое определяется как

$$P = \min_{i, r} \{L_{kl}(A_{hi}, A_{hir})\}, P \rightarrow B = \{b_{i1}, b_{i2}, \dots, b_{ir1}, \dots\}. \quad (8.7)$$

В общем случае элементы (выполняемые операции)

множества B могут быть связаны между собой следующими условиями:

1) операции различные, являются антиподами и выполняются за один и тот же промежуток времени, т. е.

$$b_i = \bar{b}_j \text{ при } i \neq j, t_i = t_j, \quad (8.8)$$

где $t_i(t_j)$ — момент времени выполнения $i(j)$ -й операции (индексы при b для упрощения записи изменены);

2) операции одинаковые и выполняются в один и тот же промежуток времени над одними и теми же операндами, т. е.

$$b_i = b_j \text{ при } i \neq j, t_i = t_j, O_i = O_j; \quad (8.9)$$

3) операции одинаковые, но выполняются в разные промежутки времени над одними и теми же операндами или в одни и те же промежутки времени, но над различными операндами, т. е.

$$\begin{aligned} b_i = b_j \text{ при } i \neq j, t_i \neq t_j, O_i = O_j \\ \text{или } b_i = b_j \text{ при } i \neq j, t_i = t_j, O_i \neq O_j; \end{aligned} \quad (8.10)$$

4) операции различные и выполняются в разные промежутки времени, т. е.

$$b_i = b_j \text{ при } i \neq j, t_i = t_j; \quad (8.11)$$

5) операции различные, но выполняются в один и тот же промежуток времени, т. е.

$$b_i \neq b_j \text{ при } i \neq j, t_i = t_j; \quad (8.12)$$

6) операции-антиподы, но выполняются в разные промежутки времени, т. е.

$$b_i = \bar{b}_j \text{ при } i \neq j, t_i \neq t_j. \quad (8.13)$$

Анализ условий (8.7) — (8.13) показывает, что совмещение ЦС и ЦА может быть проведено по следующим пяти основным направлениям.

Первое направление (метод устранения операции-антипода) предполагает минимизацию числа элементов множества B за счет устранения избыточности вида (8.8). Практически это обеспечивается сохранением базовых (основных) элементов b_i и выведением из-под знака множества вспомогательных элементов (антиподов) \bar{b}_j . Например, алгоритмы формирования периодических и квазипериодических сигналов содержат опера-

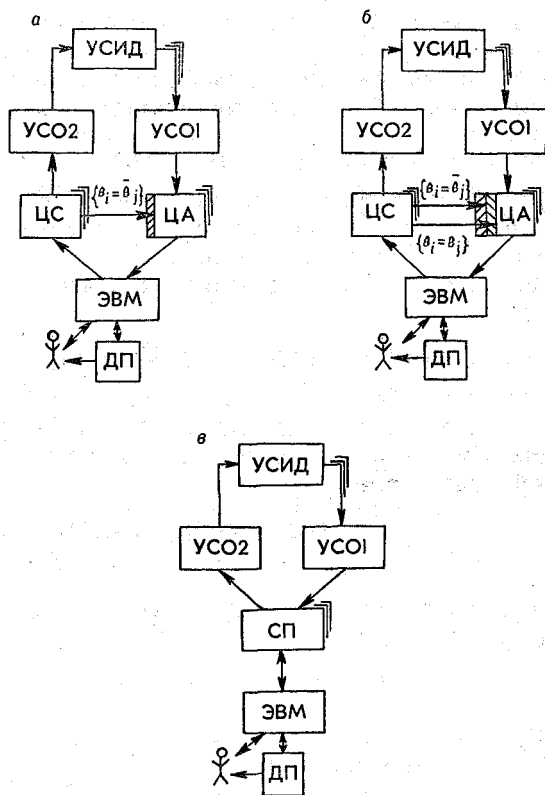


Рис. 8.12

ции формирования (b_i) кода мгновенной частоты, а алгоритмы анализа сигналов указанного класса — операции (b_j) изменения периода и формирования меток времени. Очевидно, что совместная реализация ЦС и ЦА предоставляет возможность синхронизации работы блоков синтеза и анализа (рис. 8.12, а), а это в свою очередь позволяет, во-первых, вывести из состава ЦА цифровой измеритель периода и управляемый делитель частоты, а во-вторых, минимизировать погрешность синхронизации. В АСКИ на случайные воздействия при совместной реализации алгоритмов синтеза и анализа из состава ЦА могут быть выведены средства выбора (задания) интервала анализа и шага дискретизации.

Второе направление (метод параллельного

использования средств) связано с минимизацией избыточности вида (8.9) за счет параллельного использования для синтеза и анализа (а следовательно, и устранения дублирования) необходимых технических и программных средств, исходных данных и промежуточных результатов (рис. 8.12, б). В частности, при создании АСКИ на гармонические воздействия общими в ЦС и ЦА могут быть тактовый генератор, регистр кода частоты, генератор гармонических функций, а в АСКИ на полигармонические и случайные воздействия — тактовый генератор, генератор поворачивающих множителей, устройство адресации, в программном обеспечении — программные модули, обеспечивающие одновременную работу аналогичных устройств.

Третье направление (метод последовательной реализации эквивалентных операций) определено стремлением минимизировать избыточность вида (8.10) за счет использования специализированных средств в режиме разделения времени. Это становится возможным в том случае, если предельная производительность используемых технических средств выше минимально необходимой для реализации однопрограммного режима индивидуального пользования. Иногда избыточность (8.10) удается уменьшить за счет введения дискретных элементов задержки (например, буферных регистров) результатов выполнения операции на время $t_j - t_i$ (здесь полагаем, что $j > i$).

Третье направление совмещения пары функциональных модулей полностью устраняет границы между средствами синтеза и анализа, которые при реализации первых двух направлений еще можно было провести, т. е. ЦС и ЦА сливаются в специализированный процессор СП (рис. 8.12, в). Метод последовательной реализации одинаковых операций весьма эффективен при создании СП для проведения испытаний на широкополосную случайную вибрацию, где преимущественно (45—70 % от общей трудоемкости) используются алгоритмы обратного быстрого преобразования Фурье (синтез) и быстрого преобразования Фурье (анализ), различающиеся знаком поворачивающих множителей. Примером реализации метода может служить фиксированное распределение памяти данных, запись в ОЗУ промежуточных результатов для избежания повторных расчетов и др.

В ряде случаев при практической реализации специализированных устройств АСКИ соотношения (8.10) могут

быть приведены к (8.9), когда условие $t_i = t_j$ не является жестким. Такой подход, например, при создании устройств отображения для АСКИ позволил существенно упростить устройство управления дисплейным процессором и устранить аппаратную избыточность.

Четвертое направление (метод последовательного выполнения различных операций) предполагает реализацию операций (8.11) в режиме разделения времени на технических средствах широкого применения (при наличии запаса по производительности). Например, при создании СП операция извлечения корня квадратного, используемая только при анализе, может быть реализована на универсальном арифметико-логическом устройстве, обеспечивающем выполнение основных арифметических операций в канале синтеза и анализа.

Пятое направление (метод параллельного выполнения различных операций) ориентировано на реализацию одного из известных способов параллельной обработки сигналов на различных технических средствах (8.12). При создании СП АСКИ реализация метода позволяет снизить требования к быстродействию используемых элементов, а следовательно, применять недорогие микросхемы, например последовательный умножитель 1802BP2 и др.

Минимизация избыточности вида (8.13) возможна лишь в том случае, если условие (8.13) может быть сведено к (8.8), (8.10) либо (8.11). В противном случае операции b_i и b_j реализуются отдельно специализированными средствами.

При создании специализированных устройств АСКИ возможна совместная реализация любого количества из указанных направлений совмещения пары функциональных модулей (комбинированный метод). Именно это направление обеспечивает глобальный оптимум функции цели.

Таким образом, реализация принципа совмещения пары функциональных модулей позволяет минимизировать объем оборудования проблемно-ориентированных АСКИ без ухудшения их технических характеристик, причем для повышения эффективности реализации принципа необходимо осуществлять разработку таких алгоритмов синтеза и анализа, которые были бы сравнимы по сложности с известными, но характеризовались бы большей избыточностью вида (8.8) — (8.10).

Рассмотренные принципы построения АСКИ нашли

свое воплощение в микропроцессорной системе управления испытаниями АСУ-ВИ.

Система управления АСУ-ВИ. Данная система автоматизации испытаний (САИ) (рис. 8.13) обеспечивает проведение в автоматическом и автоматизированном режимах испытаний на воздействие гармонических, полигармонических и случайных вибраций.

Ядро системы, решающее задачу применением САИ как универсального средства обработки информации, представлено процессором П микроЭВМ «Электроника-60М», оперативным запоминающим устройством ОЗУ объемом $16 \text{ К} \times 16$ бит, ППЗУ емкостью $12 \text{ К} \times 16$ бит, пультом управления (ПУПР), блоком отображения БО алфавитно-цифровой и графической информации на электронно-лучевой трубке, графопостроителем Г. Применение процессоров гармонической ПГВ и случайной ПСВ вибрации, специализированной операционной системы и пакета прикладных программ придает системе проблемно-ориентированный характер.

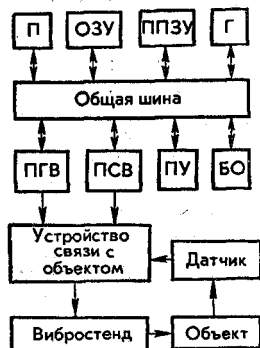


Рис. 8.13

Техническая характеристика САИ

Режим работы:

- 1) испытание методом фиксированной частоты
- 2) испытание методом качающейся частоты
- 3) испытание полигармонической вибрацией
- 4) испытание узкополосной случайной вибрацией
- 5) испытание широкополосной случайной вибрацией
- 6) снятие амплитудно-частотных и фазочастотных характеристик
- 7) спектральный анализ
- 8) панорамный спектральный анализ

Для режимов работы по пп. 1, 2, 6, 8:
 максимальный частотный диапазон, Гц
 дискретность установки границ частотного диапазона, Гц

5—10000

0,25

дискретность изменения частоты внутри диапазона, Гц	$2^{-8} - 2^{-24}$
коэффициент гармоник, %, не более	0,4
закон развертки частоты	Линейный или логарифмический
диапазон изменения скоростей развертки:	
Гц/с	0—16
окт/с	0—1
дискретность задания скорости развертки:	
Гц/с	2^{-12}
окт/с	2^{-16}
обрабатываемые и контролируемые параметры нагрузочного режима	Амплитуда первой гармоники виброускорения, действующее значение виброускорения и значение выходного тока усилителя мощности
выбор контролируемого параметра	Автоматический по заданной программе
период дискретности управления, мс, не более	10
Для режимов работы по пп. 3, 4, 5, 7:	
частотный диапазон, Гц	До 5000 при четырех отсчетах на верхней частоте и до 10000 при 2,5 отсчета на верхней частоте с размещением 512 линий в режиме задания и отображения и 1024 линии в режиме управления
число поддиапазонов	5
обрабатываемые и контролируемые параметры нагрузочного режима	Спектральная плотность мощности, действующее значение ускорения
динамический диапазон АРУ, дБ	80
погрешность управления, дБ	0,5
способ регистрации	Вывод на графопостроитель и на экран электронно-лучевой трубки
форма представления информации	Алфавитно-цифровая и графическая
масштаб отображения графиков по осям абсцисс и ординат	Линейный или логарифмический
выход на режим и сброс нагрузки	Плавный, по заданной программе

Отличительными особенностями САИ являются функционально-модульная структура и иерархический принцип организации аппаратных и программных средств,

адаптивный язык взаимодействия с системой, широкие функциональные возможности, оперативная визуализация контролируемых параметров, автоматическое документирование испытаний, современная элементная база (МПК БИС КР1802, К1804), программные средства диагностирования модулей и обнаружения ошибок оператора. В случае некорректной ситуации, угрожающей целостности объекта, предусмотрен режим аварийного сброса нагрузки.

Развитые сервисные службы организуют рациональные методы общения пользователя с системой, предоставляют возможность участия оператора в формировании условий задачи и оперативного воздействия на ход испытаний, обеспечивая диалог с оператором как в режиме настройки системы (взаимодействие по инициативе оператора), так и в режиме управления экспериментом (динамический диалог — сообщения о ходе испытаний и обработка запросов оператора, автоматическое построение графиков по опорным точкам с использованием линейной и ступенчатой аппроксимации, формирование окна на экране с оцифровкой графических зависимостей в реальных единицах, цифровая индикация выбранных маркером значений графика, изменение масштаба графика и выдача символьных сообщений).

Модульная организация системы на базе «Общей шины» позволяет расширить функциональные возможности, в частности, вводить блоки, организующие управление другими видами испытаний (на транспортные нагрузки, виброудар, случайные ударные воздействия при совмещенном воздействии климатических факторов и др.).

8.7. Системы автоматизации термоэлектро- и климатических испытаний

Информационно-измерительная система для автоматизации термоэлектроиспытаний (ИИС-ТЭ). Система предназначена для автоматизации технологического процесса проведения термоэлектроиспытаний металлических межсоединений интегральных схем. Она осуществляет изменение и регистрацию активного сопротивления межсоединений интегральных схем в процессе теплового и электрического воздействия на них и последующую статистическую обработку результатов испытаний. ИИС-ТЭ (рис. 8.14) ориентирована на использование микроЭВМ «Электроника-60» и включает процессор М2,

устройство сопряжения В1, терминал, НГМД, устройство аналогового ввода АЦП, устройство параллельного обмена ИРПР, графопостроитель, электрическую пишущую машину ЭПМ. «Consul-260», ППЗУ и источник постоянного напряжения. Специализированные модули: устройство нормализации УН, устройство коммутации

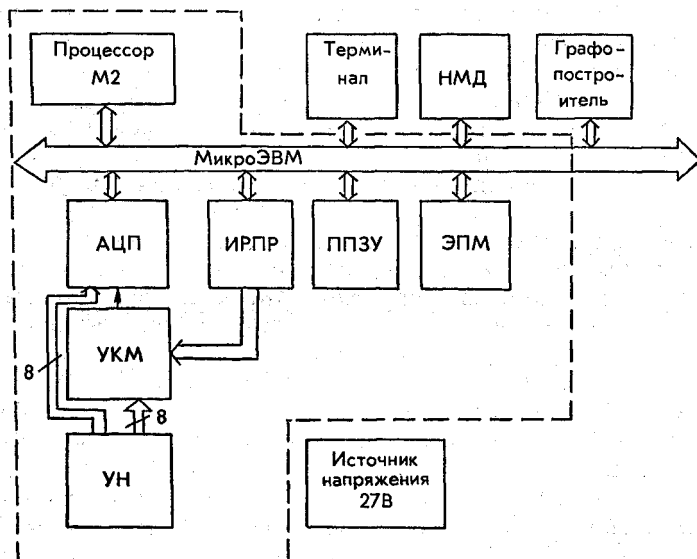


Рис. 8.14

и масштабирования УКМ. Система работает под управлением пакета прикладных программ ТЕРМО. ППП ТЕРМО построен по модульному принципу и имеет иерархическую структуру (рис. 8.15).

Техническая характеристика ИИС-ТЭ

Количество измерительных каналов, шт	8
Диапазон измеряемых сопротивлений, Ом	5—35000
Погрешность измерения сопротивлений, %, не более	1,5
Напряжение источника питания, В (Гц)	220(50)
Потребляемая мощность, Вт, не более	150

Автоматизированная система управления климатическими испытаниями АСУ «КЛИМАТ». В настоящее время

в промышленности эксплуатируется большой парк оборудования для проведения климатических испытаний на воздействие тепла, холода, влаги, давления. Значительная часть этого оборудования (термо-, баро-, влагокамеры) имеет лишь примитивные режимы автоматического управления. При постоянном повышении качества испы-

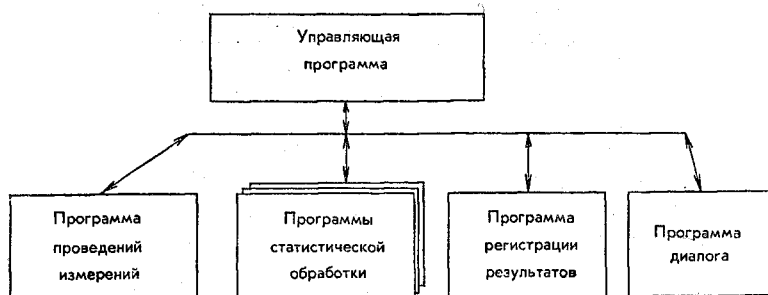


Рис. 8.15

таний и увеличении эффективности труда испытателя необходимо обеспечить соответствующую модернизацию существующей техники такого класса.

Решению этих задач служит автоматизированная система управления климатическими испытаниями АСУ «КЛИМАТ», созданная на базе микроЭВМ ДВК-2М (ДВК-3). Система предназначена для автоматизации испытательных камер тепла, холода, влаги, давления типов КТК, РЕВТРОН, ТВ, ТВВ. Она обеспечивает управление исполнительными устройствами камеры, контроль фактического состояния этих устройств, измерение параметров (температуры, влажности, давления) в различных точках камеры и на испытуемом изделии с помощью датчиков — терморезисторов или термопар. МикроЭВМ с набором периферийных устройств обеспечивает ведение процесса испытаний по заданной программе, обработку и документирование результатов, диалог с испытателем.

Техническая характеристика АСУ «КЛИМАТ»

Количество каналов управления исполнительными устройствами	12
Количество контролируемых исполнительных устройств	24
Количество входов для подключения терморезисторов	12

Количество входов для подключения термопар	20
Объем ОЗУ данных с батарейной подпиткой, К байт	64
Период дискретизации цикла управления, с	1
Количество одновременно обслуживаемых термокамер	До 16
Разрядность АЦП	12
Погрешность регулирования, град	0,3

Автоматизированная система управления климатическими испытаниями АСУ «КЛИМАТ-01» представлена на рис. 8.16. В ней используется принцип централизованного управления. Система состоит из микроЭВМ ДВК-2М, контроллера интерфейса КИ и контроллера термокамеры КТ. МикроЭВМ и КИ используют связь с помощью системной магистрали «Общая шина», а контроллеры между собой соединены синхронным последовательным интерфейсом с гальванической развязкой.

МикроЭВМ содержит процессор, НГМД, устройство отображения УО, устройство документирования УД и выполняет все вычислительные и сервисные операции. В состав КИ входят ОЗУ данных с батарейной подпиткой ОЗУБ, таймер, блок связи с климатическими камерами (БСК), блок сопряжения с «общей шиной» микроЭВМ БСОШ. ОЗУБ имеет страничную организацию и предназначено для хранения результатов испытаний и промежуточных данных и защищено от потери информации при сбойных ситуациях по цепям питания. Применение такого ОЗУ вызвано большой длительностью испытаний и значительной стоимостью испытываемых изделий сложной техники, которые могут прийти в негодность вследствие сбойных ситуаций.

Невысокая точность встроенного в микроЭВМ таймера требует реализации такого устройства в виде дополнительного блока на основе генератора с кварцевой стабилизацией. БСК реализует обмен с контроллером термокамеры по синхронному последовательному принципу, что обеспечивает повышенную надежность и простоту реализации. К недостатку следует отнести увеличенное количество связей — 6 витых пар проводов.

Контроллер термокамеры осуществляет прием управляющей информации от микроЭВМ и выдает ее в термокамеру через усилители мощности УМ с гальванической развязкой (оптотиристоры, оптотранзисторы), прием информации от токовых датчиков состояния исполнительных устройств ДСУ, преобразование с помощью

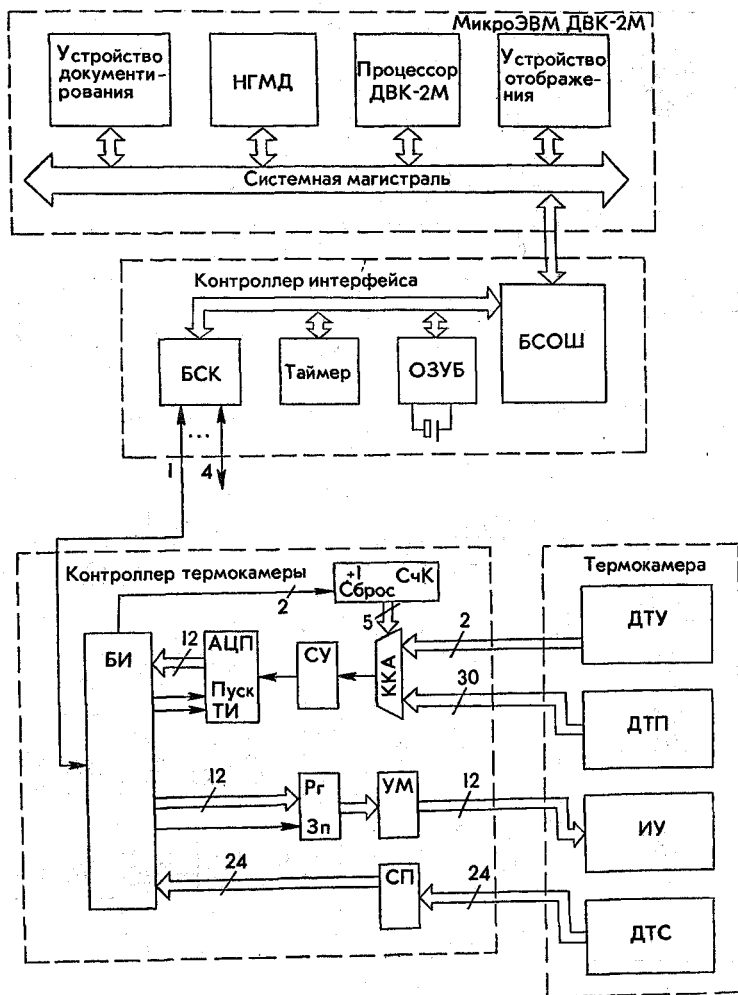


Рис. 8.16

согласователей преобразователей СП и передачу ее в микроЭВМ для контроля и диагностирования, аналого-цифровое преобразование информации, поступающей через согласующие усилители СУ от датчиков температуры.

Освоение отечественной промышленностью однокристальных ЭВМ дает новые возможности в реализации

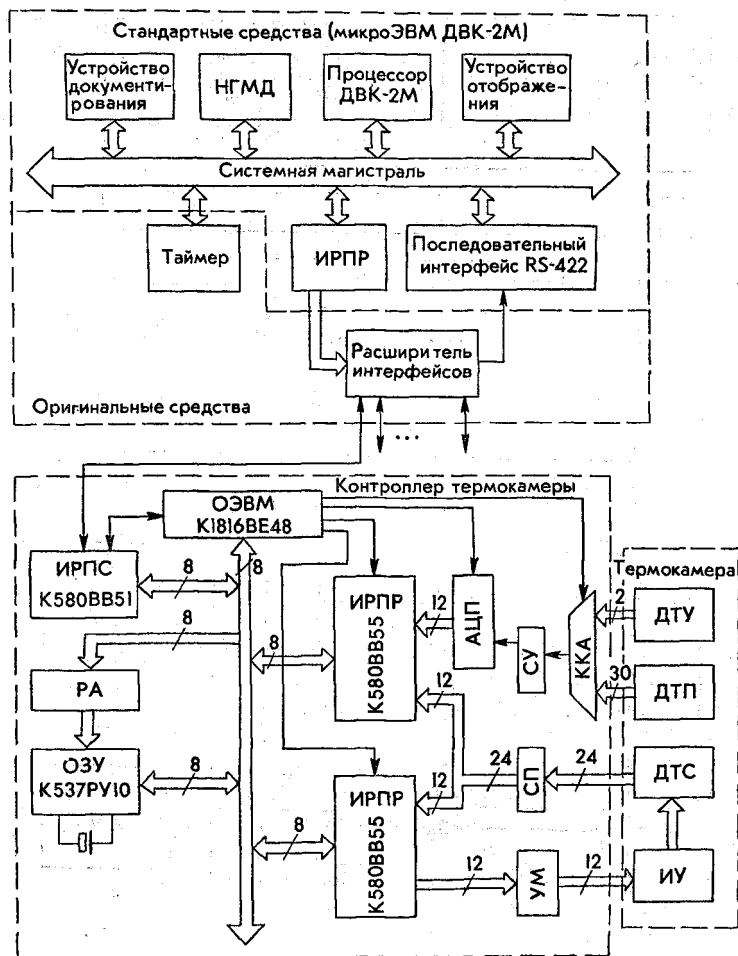


Рис. 8.17

различного рода контроллеров, в том числе и для автоматизации климатических испытаний.

Развитием рассмотренной АСУ «КЛИМАТ-01» является система «КЛИМАТ-02» (рис. 8.17). В ней используется двухуровневый принцип построения. Верхний уровень образует микроЭВМ ДВК-2М (ДВК-3) со стандартным набором средств (процессор, НГМД, УО, УД, ИРПР, ИРПС) и минимальным объемом оригиналь-

ных блоков (таймер, расширитель интерфейсов). При этом, если требования к точности таймера невысоки, используется внутренний таймер. При небольшом количестве обслуживаемых камер (одна-две) нет необходимости в расширителе интерфейсов и блоке параллельного интерфейса для его управления.

Нижний уровень (контроллер термокамеры) реализован на микросхемах большой и сверхбольшой степени интеграции. Ядром контроллера является однокристалльная ЭВМ K1816BE48. К ОЭВМ подключены интерфейсные БИС, ОЗУ. Интерфейсные БИС обслуживают связь с микроЭВМ, АЦП, усилителем мощности УМ, согласователем-преобразователем СП. В данной структуре используется унифицированный последовательный интерфейс с гальванической развязкой стык С2. Верхний уровень управления обеспечивает диалог с испытателем (подготовка программ испытаний, выдача различных справок), документирование результатов испытаний, загрузку программ испытаний в соответствующий контроллер термокамеры, прием данных из контроллера. Нижний уровень осуществляет ведение программы испытаний, контроль состояния термокамеры, обмен данными с микроЭВМ. Такая организация позволяет резко увеличить надежность испытаний (особенно для случая большого количества одновременно обслуживаемых камер) за счет автономии нижнего уровня управления, разгрузить микроЭВМ, повысить эффективность и удобство работы оператора-испытателя.

Программное обеспечение АСУ функционирует в среде ОС RT-11SJ. ПО реализовано по модульному принципу на языке ПАСКАЛЬ с элементами МАКРОАССЕМБЛЕРА и содержит модули поддержки диалога с оператором, документирования, обработки, режима реального времени.

Контрольные вопросы

1. Приведите классификацию систем управления промышленными роботами.
2. Назовите способы построения многопроцессорных систем управления роботами.
3. Какие компоненты включает в себя программное обеспечение роботов?
4. На базе каких микропроцессоров и микроЭВМ реализованы системы управления «Гранит-02», «Сфера-36», 2С42, «Электроника НЦ-31»?
5. Какие системы автоматизации испытаний вы знаете?

ОТВЕТЫ

Глава 2

1. Наибольшей значащей десятичной цифрой. 2. Основная структурная единица информации в ЭВМ. 3. $1111_{(2)}$, поскольку это число равно $15_{(10)}$. 4. Деление. 5. а) 10111; б) 1101001; в) 100000; г) 1111; д) 11001110; е) 10000000; ж) 111111; з) 11101; и) 1100.001; к) 10000.011; л) 10.1.

6. Двоичные Восьмеричные Шестнадцатеричные:

а) 01111110	176	7E
б) 100	4	4
в) 10000	20	10
г) 111111	77	3F
д) 1100101	145	65
е) 1100	14	C
ж) 1	1	1
з) 01111111	177	7F
и) 1001.01	11.2	9.4

7. $2^n - 1 \leq X \leq 0$.

8. Прямой: Обратный: Дополнительный:

а) 1.1101101	1.0010010	1.0010011
б) 0.1110110	0.1110110	0.1110110
в) 1.00001	1.11110	1.11111
г) 1.111011	1.000100	1.000101

9. 00.10001. 10. 11.1100. 11. 10.11000. 12. 00.1010. 13. 10.10001. 14. 1.00010. 15. 1.10000. 16. 1.10011. 17. 0.11.

Глава 3

1. $y = x_1(x_2 + \bar{x}_2) + \bar{x}_1x_2 = x_1x_2 + x_1\bar{x}_2 + \bar{x}_1x_2$. 2. $y = x_1\bar{x}_3$; склеивание по x_2 . 3. $y = x_1x_2(1 + x_3x_4) = x_1x_2 \cdot 1 = x_1x_2$. 4. $y = x_1x_2 + x_3x_4 = x_1x_2 + \bar{x}_3x_4$. Получилась ДНФ. 5. $y = x_1x_2 + x_1x_3 + x_2x_3$. 6. $y = x_1x_2x_3x_4$. 7. $y = (x_1 + x_2) + (x_1 + x_2 + x_4)$. 8. $y_{\min} = x_1x_2 + x_2\bar{x}_3 + \bar{x}_2x_3$ либо $y_{\min} = x_1x_3 + x_2x_3 + x_2\bar{x}_3$. 9. В автомате Мили выходной сигнал зависит от входного сигнала и состояния автомата, а в автомате Мура — только от состояния автомата.

Глава 4

1. а) $U_y = 0,9$ В. Сначала определяем напряжение U_d в средней точке делителя при отключенных диодах (рис. 1, а): $U_d = \frac{6 \cdot 2R}{R + 2R} = 4$ В. Проверяем, у какого диода наибольшая разность потенциалов

в отпирающем направлении (напряжение на аноде U_A больше напряжения на катоде U_K). Для $VD1$ $U_{a,k} = U_a - U_k = U_n - U_{x1} = 4 - 0,2 = 3,8$ В. Для $VD2$ $U_{a,k} = U_d - U_{x2} = 4 - 3 = 1$ В. Так как у $VD1$ эта разность больше и при этом $U_{a,k} > U_{d,з} = 0,6$ В, то первым

подключаем $VD1$ (рис. 1, б). Диод $VD1$ открывается, напряжение на выходе $U_y = U_{x1} + U_{д.01} = 0,2 + 0,7 = 0,9$ В.

Подключаем диод $VD2$ (рис. 1, в) и убеждаемся, что $VD2$ закрыт, так как напряжение на катоде 3 В, больше напряжения на аноде 0,9 В; подключение $VD2$ не изменяет выходного напряжения $U_y = 0,9$ В. б) 3,7 В; в) 4 В, так как у обоих диодов разность $U_d - U_k < U_{д,з} = 0,6$ В, оба диода закрыты, $U_y = U_d = 4$ В; г) 4 В (диоды закрыты); д) $VD1$ — открыт, $VD2$ — закрыт, $U_y = -2 + 0,7 = -1,3$ В; е) оба диода открыты, $U_y = 0,4 + 0,7 = 1,1$ В.

2. а) $U_y = -0,9$ В. По аналогии с (4.1) $U_d = -2,5$ В; $U_{x1} - U_d = -0,2 - (-2,5) = 2,3$ В $> U_{д,з} = 0,6$ В, $U_{x2} - U_d = -3,6 - (-2,5) = -1,1$ В. Открыт $VD1$, закрыт $VD2$; $U_y = U_{x1} - U_{д.01} = -0,2 - 0,7 = -0,9$ В; б) оба диода $VD1$ и $VD2$ открыты, $U_y = -0,9$ В; в) оба диода закрыты, так как $U_{a,k} < 0,6$ В, $U_y = -2,5$ В; г) открыт $VD1$, закрыт $VD2$, $U_y = 2,4 - 0,7 = 1,7$ В; д) оба диода закрыты, $U_y = -2,5$ В.

3. $U_{пор}^0 = U_{э,з}(1 + R1/R2) = 0,6(1 + 1) = 1,2$ В.

$$4. t_{\Phi}^+ = 3R_K C_H = 3 \cdot 10^3 \cdot 200 \cdot 10^{-12} = 600 \cdot 10^{-9} \text{ с} = 600 \text{ нс}; t_{\Phi}^- = \frac{E C_H}{\beta[(U^1 - U_{э,0})/R1 - U_{э,0}/R2]} = \frac{E C_H}{\beta[(U^1 - 2U_{э,0})/R]} = \frac{5 \cdot 200 \cdot 10^{-12}}{50[(2,4 - 2 \cdot 0,2)/10^4]} = 100 \text{ нс}; t_{\Phi}^+ / t_{\Phi}^- = 600/100 = 6.$$

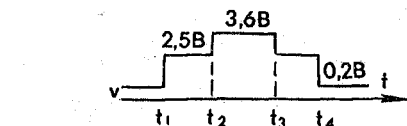


Рис. 2

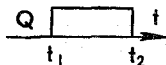


Рис. 3

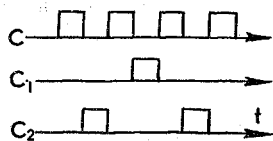


Рис. 4

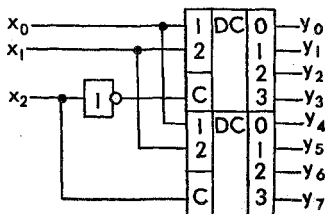


Рис. 5

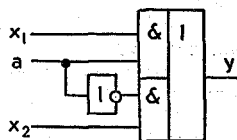


Рис. 6

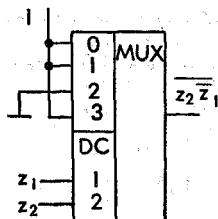


Рис. 7

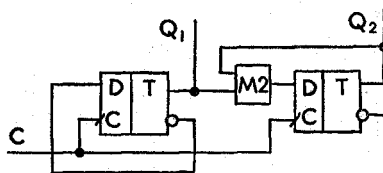


Рис. 8

5. $U_{пор}^0 = U_{э.35} + U_{а31} - U_{д.ш.03} = 0,6 + 0,6 - 0,4 = 0,8$ В; $U_n^+ = U_{пор}^0 - U_{вх}^0 = 0,8 - 0,5 = 0,3$ В; $U_{пор}^1 = U_{э.05} + U_{э.01} - U_{д.ш.33} = 0,7 + 0,7 - 0,3 = 1,1$ В; $U_n^- = U_{вх}^1 - U_{пор}^1 = 2,7 - 1,1 = 1,6$ В.
6. См. рис. 2. 7. См. рис. 3. 8. См. рис. 4. 9. См. рис. 5. 10. $y = ax_1 + \bar{a}x_0$ (см. рис. 6). 11. См. рис. 7. 12. См. рис. 8.

Глава 6

1. Подсистема аналогового ввода (см. рис. 6.2). 2. Подсистема аналогового вывода без СУ (см. рис. 6.2). 3. См. рис. 6.5. 4. 16 МГц. 5. $Q = 1920$ ячеек. $T_{озур} = 64(1 - 0,2)/80 = 0,64$ мкс. К541РУ1, К537РУ2, К537РУ3, К565РУ1 и т. п. 6. Последовательно нажать клавиши R, 0, 2, 0, 0, 0. 7. $N = 10 \cdot 32 \cdot 10^6/8 = 4 \cdot 10^7$; где 10 — число метров; $32 \cdot 10^6$ — плотность записи, бит/м; 8 — число двоичных разрядов кода буквы. 8. Время — $1/300$ мин = $0,02$ с. Объем информации — $0,02$ с \times 250 К бит/с = 5 К бит. 9. Время — $(60 \times 9)/900 = 0,6$ с, где 9 — число ударов для вывода одной буквы; 900 — максимальная тактовая частота срабатывания иглы, удар/сек. 10. $20 \times 4/40 = 2$ с, где 40 — максимальная скорость движения, см/с; 4 — число вычерчиваемых линий.

Глава 7

1. Пусть программы располагаются с адреса 1000.
1000) 12700
1002) 100 В R0 заносится начальный адрес
1004) 12701
1006) 200 В R1 заносится конечный адрес
1010) 5020 Обнуление ячейки по адресу R0 с инкрементированием
1012) 20100 Сравнение текущего адреса с конечным

1014) 2375 Переход к ячейке 1010 в случае $[P1] > [R0]$
 Пусть цифры располагаются в ячейках 200, 202, 204, 206 по степени возрастания веса. Результат — в ячейке 210.

1000) 12700

1002) 200 В R0 заносится адрес младшей цифры

1004) 12701

1006) 4 В R1 заносится число цифр

1010) 12702

1012) 210 В R2 заносится адрес результата

1014) 10103 В R3 заносится число двоичных зарядов одной десятичной цифры

1016) 6210 Сдвиг очередной десятичной цифры в ячейку результата

1020) 6012

1022) 77303 Организация внутреннего цикла для сдвига четырех двоичных разрядов одной десятичной цифры (возврат к ячейке 1016)

1024) 5720 Увеличение на 2 адреса десятичной цифры

1026) 77106 Организация внешнего цикла для перебора четырех десятичных цифр

1030) 0

Пусть числа располагаются в ячейках 200, 202, 204, 206. Результат помещается в ячейку 206.

1000) 12700

1002) 200

1004) 12701

1006) 2

1010) 62020

1012) 77102

1014) 0

2. TITLE TIMER

R5 = %5

HOUR = 200; HOUR — АДРЕС ЧАСОВ

MIN = 202 MIN — АДРЕС МИНУТ

SEC = 204; SEC — АДРЕС СЕКУНД

INC #SEC

CMP 60., #SEC

BPL M1

CLR #SEC

INC #MIN

CMP 60., #MIN

BPL M1

CLR #MIN

INC #HOUR

M1: RTS R5

END

3. См. прим. 7.2 с другими адресами регистров.

4. TITLE PILA

RO = %0

RS = 160000

RD = 160002

MACRO OPROS

M1: TSTB #ARG

BGE M1

ENDM

M2: OPROS RS

```
MOV RO, #RD ; ПЕРЕСЫЛКА КОДА В ЦАП
INC RO      ; УВЕЛИЧЕНИЕ КОДА
```

```
BR M2
.END
```

5. 10 DIM A(100), B(100)
20 FOR I=1 TO 5
30 READ A(I)
40 NEXT I
50 INPUT «ВВЕДИТЕ X», X
60 INPUT «ВВЕДИТЕ Y», Y
70 J=1
80 FOR I=1 TO 5
90 IF A(I) >= X THEN IF A(I) < Y THEN B(J)=I:J=J+1
100 NEXT I
110 FOR I=1 TO 5
120 PRINT B(I),
130 NEXT I
140 DATA 10., 11, 12, 13, 14, 15
150 STOP
6. См. пример ранжирования чисел. Массивы и переменные, за исключением индексов типа «строка символов».
7. 10 INPUT «ВВЕДИТЕ A», A
20 INPUT «ВВЕДИТЕ B», B
30 INPUT «ВВЕДИТЕ C», C
40 DS = B² - 4*A*C
50 IF DS < 0 THEN PRINT «УРАВНЕНИЕ НЕ ИМЕЕТ
ДЕЙСТВИТЕЛЬНЫХ КОРНЕЙ»: GOTO 100
60 D = SQR(DS)/(2*A)
70 DD = -B/(2*A)
80 X1 = DD + D: X2 = DD - D
90 PRINT «X1=»X1; «X2=»X2
100 STOP
8. 10 INPUT «ВВЕДИТЕ ШАГ», DX
20 FOR X=0 TO 6.28 STEP DX
30 PRINT TAB (INT(COS(X)*40+40))«*»
40 NEXT X
50 STOP

ПРИЛОЖЕНИЯ

1. Список директив интерпретатора БЕЙСИК

NEW	Стирание программной памяти и памяти данных.
RESET	Перезапись буферов всех открытых файлов. После этого файлы закрываются и содержимое буферов стирается.
AUTO	Автоматическая генерация номеров строк. Выход из режима — ^C.
CLEAR	Обнуление переменных.
CONT	Продолжение прерванной программы.
SYSTEM	Выход в операционную систему. Передается управление программам ОС.
EDIT	Редактирование строки программы.
LIST и LLIST	Вывод текста программы на экран дисплея и печатающего устройства соответственно.
DELETE	Удаление строк программы.
RENUM	Автоматическое изменение номеров строк.
SAVE	Сохранение программы в виде файла на дискете.
LOAD	Загрузка файла-программы в ОЗУ.
RUN	Загрузка в ОЗУ и запуск программы.
MERGE	Присоединение к загрузочному тексту программы другого текста в машинных кодах.
FILES	Вывод на экран дисплея списка файлов дискеты.
KILL	Стирание файлов.
NAME	Переименование файлов. NAME <старое имя> AS <новое имя>.
TRON и TROFF	Включение и выключение программы трассировки.

2. Список операторов языка БЕЙСИК

Простые управляющие операторы

LET	Оператор присваивания.
REM	Комментарий к программе.
END	Конец БЕЙСИК-программы.
STOP	Прерывание выполнения программы.
DEFINT,	Операторы объявления типов данных.
DEFSNG,	Явное объявление данных целого типа,
DEFDBL,	вещественного одинарной точности, веще-
DEFSTR, DEF FN	ственного двойной точности, символьного типа и встроенной функции.

CINT, CDBL
CSNG

Преобразование данных вещественного типа в данные целого, вещественного двойной точности и вещественного одинарной точности.

Работа с массивами

DIM
OPTION BASE

Описание массива.
Определение нижней границы размерности массива.

ERASE

Изменение размерности и (или) числа элементов массива.

Условные и безусловные переходы

FOR, TO,
STEP, NEXT
WHILE, WEND

Программирование цикла с фиксированным числом шагов.
Программирование цикла с переменным числом шагов.

GOTO
IF-THEN-ELSE

Безусловный переход.
Условный переход.

Работа с подпрограммами

ON, GOSUB
RETURN
CALL

Обращение к подпрограмме.
Возврат из подпрограммы.
Вызов подпрограммы, написанной в машинном коде.

VARPTR

Установка начального адреса первой ячейки области памяти для задаваемых переменных.

DEF USR

Включение в программу подпрограммы, написанной в машинном коде.

CHAIN
COMMON

Дозагрузка программных модулей в ОЗУ.
Передача информации для сцепленных и наложенных программных модулей.

%INCLUDE

Указание имен программных блоков на языке БЕЙСИК, которые следует считывать из внешней памяти.

Операторы ввода-вывода

INPUT
LINE INPUT

Ввод с клавиатуры.
Ввод с клавиатуры переменных типа «строка символов».

PRINT

Бесформатный вывод данных на экран дисплея.

FRINT USING

Форматный вывод данных на экран дисплея.

LPRINT и LPRINT USING

Вывод данных на принтер.

WRITE
WIDTH

Вывод данных на экран дисплея.
Задание максимального числа символов в строке.

OUT

Вывод данных непосредственно в выходной порт.

POKE

Запись байта данных по явно указанному адресу ОЗУ.

WAIT

Синхронизация вычислений при работе с внешними устройствами (ожидание совпадения заданного кода-маски с информа-

SWAP цией порта ввода).
Обмен значениями между переменными
одного типа.

Работа с наборами данных

DATA READ	Организация данных в наборах. Считывание данных, организованных в наборы.
RESTORE	Настройка указателя набора данных на данные, стоящие в начале набора.
OPEN	Открытие набора данных и установка ме- тода доступа.
INPUT #, LINE INPUT # PRINT #, PRINT # USING WRITE GET	Считывание данных в коде КОИ-7, Неформатный и форматный вывод дан- ных в последовательный набор. Запись информации в набор данных. Пересылка записи из внешней памяти в область ОЗУ.
PUT	Пересылка заданной области ОЗУ во внеш- нюю память.
FIELD	Установка структуры записи для файлов с прямым доступом.
LSET и RSET	Добавление значений выражений из строк символов к левоустановленным или пра- воустановленным текстам.
CLOSE	Пересылка текущего содержимого области ОЗУ во внешнюю память.

3. Список встроенных функций транслятора БЕЙСИК

Ввод и вывод на языке БЕЙСИК

INKEY	Установка пустой строки символов, если не была нажата ни одна из клавиш.
INPUT	Ввод переменной типа «строка символов» с заданным числом символов.
INP	Чтение байта данных непосредственно из порта.
PEEK	Чтение байта из ОЗУ по непосредственно заданному адресу.
SPC	Задание нужного числа пробелов.
TAB	Установка курсора в заданную позицию строки.
LPOS	Установка позиции печатающей головки принтера.
POS	Определение номера текущей позиции курсора.

Функции для работы с наборами данных

EOF	Установка признака, если имеет место условие конца набора данных.
LOF	Указание числа записей в последнем считанном или записанном на внешнюю память блоке.

LOC

MKI α , MKS α ,
MKD α

CVI, CVS, CVD

Указание номера записи, к которой происходил последний доступ при работе с каким-либо набором данных прямого доступа.

Преобразование данных целого типа, вещественного одинарной и двойной точности соответственно в переменные типа «строка символов».

Преобразование данных типа «строка символов» длиной 2, 4 и 8 байтов соответственно в переменные целого, вещественного типа одинарной и двойной точности.

Работа с символьными строками

CHR α

Преобразование значения целочисленного выражения в соответствующие символы кода КОИ-7.

ASC

Преобразование первого символа выражения типа «строка символов» в код КОИ-7.

STR α

Преобразование значения числового выражения в строку символов.

VAL

Преобразование строки символов в числовое выражение.

HEX α

Преобразование десятичного значения числа в шестнадцатеричное.

OCT α

Преобразование десятичного значения числа в восьмеричное.

RIGHT α

Формирование подстроки символов справа.

LEFT α

Формирование подстроки символов слева.

SPACE α

Формирование строки символов, состоящей из пробелов.

STRING α

Формирование строки из какого-либо предварительно заданного символа, который добавляется N раз.

INSTR

Определение строки символов в другой строке.

MID α

Выбор подстроки символов из какой-либо строки.

LEN

Определение длины, т. е. количества символов в строке.

Стандартные математические функции

ABS

Вычисление модуля переменной.

FIX

Определение целой части.

INT

Округление.

SGN

Определение знака.

SQR

Вычисление квадратного корня.

EXP

Вычисление экспоненциальной функции.

LOG

Вычисление логарифма.

SIN

Вычисление синуса.

COS

Вычисление косинуса.

TAN

Вычисление тангенса.

ATN

Вычисление арктангенса.

RND

Вычисление псевдослучайного числа.

FRE

Указание количества доступного места в ОЗУ.

ЛИТЕРАТУРА

- Балашов Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы: Учеб. пособие для вузов/Под ред. В. Б. Смоллова.— М.: Радио и связь, 1981.— 328 с.
- Брябрин В. М. Программное обеспечение персональных ЭВМ.— М.: Наука, 1988.— 272 с.
- Вигдорчик Г. В. и др. Основы программирования на АССЕМБЛЕРЕ для СМ ЭВМ/Г. В. Вигдорчик, А. Ю. Воробьев, В. Д. Праченко: Под общ. ред. В. П. Семика.— М.: Финансы и статистика, 1987. 240 с.
- Глушков В. М. Основы безбумажной информатики.— М.: Наука, 1987.— 552 с.
- Дьяконов В. П. Справочник по алгоритмам на языке БЕЙСИК для персональных ЭВМ.— М.: Наука, 1987.— 240 с.
- Иванов Е. Л. и др. Периферийные устройства ЭВМ и систем: Учеб. пособие для вузов/Е. Л. Иванов, И. М. Степанов, К. С. Хомяков.— М.: Высш. шк., 1987.— 319 с.
- Каган Б. М., Сташин В. В. Основы проектирования микропроцессорных устройств автоматики.— М.: Энергоатомиздат, 1987.— 304 с.
- Лысков Б. Г. Арифметические и логические основы цифровых автоматов/Учебник для вузов.— Мн.: Высш. шк., 1980.— 336 с.
- Майоров С. А., Новиков Г. И. Структура электронных вычислительных машин.— Л.: Машиностроение, 1979.— 384 с.
- МикроЭВМ/Пер. с англ. под ред. А. Дирксена.— М.: Энергоиздат, 1982.— 328 с.
- Микропроцессоры/Под ред. Д. Л. Преснухина.— М.: Высш. шк. В 3 кн. 1986—1987.
- Мини- и микроЭВМ семейства «Электроника»/Б. Л. Толстых, И. Л. Талов и др.— М.: Радио и связь, 1987.— 296 с.
- Пешков А. Т. Периферийные устройства цифровых ЭВМ: Учеб. пособие для вузов.— Мн.: Высш. шк., 1980.— 256 с.
- Применение управляющих вычислительных машин: Учеб. пособие/А. Н. Морозевич, А. В. Николаев, А. П. Пашкевич, А. А. Петровский.— Мн.: Высш. шк., 1988.— 238 с.
- Смоляров А. М. Системы отображения информации и инженерная психология: Учеб. пособие.— М.: Высш. шк., 1982.— 272 с.
- Современные методы и устройства отображения информации/Под ред. М. И. Кривошеева и А. Я. Брейтбарта.— М.: Радио и связь, 1981.— 216 с.
- Соловейчик И. Е. Дисплеи в системах с ЭВМ.— М.: Сов. радио, 1979.— 211 с.
- Строганов Р. П. Управляющие машины и их применение: Учеб. пособие.— М.: Высш. шк., 1986.— 240 с.
- Уэйт М., Ангермайер Дж. Операционная система CP/M: Пер. с англ.— М.: Радио и связь, 1986.— 312 с.
- Фоли Дж., Вэн Дэм А. Основы интерактивной машинной графики: В 2-х кн. Пер. с англ.— М.: Мир, 1985.— 368 с.
- Франке К. Введение в микроЭВМ: Пер. с нем.— М.: Энергоатомиздат, 1988.— 160 с.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ*

- Автомат абстрактный 67
— конечный 67
— Мили 68, 70, 71
— Мура 68, 70, 72
— структурный 67
Автоматизация программирования 12, 247, 248
Адрес возврата 177
— памяти 149
— слова 13
Адресация косвенная 242
— непосредственная 242
— прямая 242
— регистровая 242
Алгебра логики (булева) 46
Алгоритм 13, 171, 223
Алфавит 251, 259
Архитектура микропроцессора (микроЭВМ) 141
АССЕМБЛЕР 233, 250

Байт 30, 149, 225
— состояния 144, 147
Банк памяти 166
Бит 26, 30, 225

Ввод 167, 170, 179
Вес разряда 19
Выборка однокоординатная 133
— двухкоординатная 133
Вывод 169, 170, 180

Головка магнитная 208
Графопостроитель барабанного типа 220
— планшетного типа 220

Демультимплексор 106, 108
Дешифратор одноступенчатый (линейный) 108
— стробируемый 108
Диаграмма Вейча 63—65
Дизъюнкция 48, 52, 53
Диод Шотки 78, 79
Домен цилиндрический магнитный 214

Доступ независимый 201
— последовательный 202
— прямой 202
Драйверы периферийных устройств 270

Задержка распространения сигнала 77
Закон инверсии (отрицания) 53, 59
— переместительный 52
— распределительный 53
— сочетательный 52

Импlicants простая 57
Инвертор простой 80, 83
— сложный 80
Интерпретатор 248, 249, 259
Интерфейс 167, 169, 288
ИСКЛЮЧАЮЩЕЕ ИЛИ 50, 51
ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ 51, 52

Карта Карно 60, 63—65, 115, 126, 128
Клавиатура универсальная алфавитно-цифровая 204, 206
— функциональная 207
Код числа 30
— — дополнительный 31
— — обратный 31
— — прямой 30
Команда 13, 225, 271
— арифметическая 226, 227, 246
— безусловного перехода 247
— ветвления 226, 239
— логическая 226, 227, 247
— пересылки данных 226
— резидентная 271, 272
— транзитная 271, 273
— управления 226, 240, 247
— управляющих символов 271, 274
— условного перехода 239, 247
Компаратор (схема сравнения) 106, 113

* Составил А. Н. Морозевич.

Компилятор 248, 259
Конstituента единицы 56
— нуля 58
Контроллер 10, 150, 169, 170,
174, 313
Конъюнкция 47, 52, 53
— элементная 55, 56
Коэффициент объединения 78
— разветвления 78

Логика отрицательная (ОЛ)
76, 80
— положительная (ПЛ) 76, 80

Магистраль двунаправленная
143, 144

МАКРОАССЕМБЛЕР 250

Макрокоманда 250, 256

Масштаб времени реальный 8

Матрица (таблица) импликант-
ная 62, 63

— логическая программируе-
мая (ПЛМ) 136

Машина вычислительная управ-
ляющая 7

— — электронная 5, 12

Метод адресации автодекремент-
ный 233, 235

— — автоинкрементный 233,
234

— — индексный 233, 236

— — косвенно - автодекре-
ментный 237

— — косвенно - автоинкре-
ментный 237

— — косвенно - индексный
237, 238

— — косвенно - регистровый
237

— — регистровый 233, 242

— Квайна 60—63

— следящий 216

— развертывающий 216

— формирования контуров
изображения 217

Микрокоманда 155, 159

Микропроцессор 10, 141, 146

— многокристалльный 141, 152

— однокристалльный 141, 142

МикроЭВМ 10, 141, 150

Модуль программный 223

Мощность динамическая 77

— средняя 76

Мультиплексирование 143, 144
Мультиплексор 109—112, 146,
154, 158

Набор переменных 46

Накопитель магнитный 208

Неравнозначность 50, 51

Носитель магнитный 208

Обеспечение программное 223,
291, 294, 315

— — общее 223

— — специальное 223

Операнд 225, 227, 250

Основание системы счисления 18,
19

Отладка 280

Отрицание дизъюнкции 50

— конъюнкции 49

— логическое 47

Память 16, 146, 163, 165

Пакет прикладных программ
224, 274

Переменная двоичная 46

Перо световое 207

Подсистема аналогового ввода
188

— — вывода 189

Полусумматор (ПСМ) 114

Помехоустойчивость 76, 77, 81

Порт ввода-вывода 150

Правило двойного отрицания 54,
59

— поглощения 54

— склеивания 54, 56, 57, 60

Программа 13, 223, 248, 296,
308

— абсолютная (загрузочная)
249

— обработки прерывания 174

— объектная 248, 249, 258

Проектирование восходящее 283
— нисходящее 283

Пространство адресное 165, 166,
241

Процессор командный 271

Прямой доступ к памяти 170,
173, 183

Равнозначность 51, 52

Регистр 117

— аккумулятор 144, 148

- Регистр адреса 144
 - косвенного 149
 - данных 169
 - общего назначения (РОН) 144, 149, 226
 - признаков 149
 - реверсивный 119
 - сдвигающий 118, 119
 - состояния 158, 169
- Связь непосредственная 80
 - резисторная 80
 - резисторно - конденсаторная 80
- Сигнал импульсный 75
 - потенциальный 75
- Система команд 225, 226, 241, 250
 - операционная 204, 268, 271, 291
 - счисления 18, 247
 - позиционная 18
 - условных обозначений
 - элементов 58
- Сложение логическое 48
 - по модулю два 50, 51
- Совместимость программная 8
- Сетка машины разрядная 24
- Способ регистрации 217
 - механический 217
 - немеханический 217
- Стек 149, 162, 175, 234, 244, 245
- Степень функциональной интеграции 74
- Структура данных 24
 - ЭВМ 15
- Сумматор 113
 - полный одnorазрядный (ПОС) 114
 - с параллельным переносом 116
 - с последовательным переносом 115
- Схема логическая 66
 - диодно-резисторная (ДРЛ) 78
 - комплементарная на МОП-транзисторах (КМОП) 91, 93
 - транзисторно - транзисторная (ТТЛ) 84, 86
 - на транзисторах Шотки (ТТЛШ) 87
 - транзисторная с непосредственными связями (НСТЛ) 80
 - — — с резисторно-конденсаторными связями (РКТЛ) 80
 - — — с резисторными связями (РСТЛ) 80
 - — эмиттерно - связанная (ЭСЛ) 87
 - сравнения 106, 113
- Счетчик 119, 121
 - вычитающий 120, 122, 123
 - программный 144
 - реверсивный 120, 125
 - суммирующий 120, 121, 123, 124
- Таблица выходов 68, 69, 70
 - истинности 46, 55, 134
 - переходов 68, 69, 70
 - отмеченная 70
- Транзистор Шотки 86
- Транслятор 248, 258, 259
- Трансляция 248
- Требование прерывания 171, 172, 175
- Триггер 75, 93
 - асинхронный 95, 105
 - двухступенчатый 95, 104, 105
 - одноступенчатый 95
 - разрешения прерываний 150
 - синхронный 95, 99, 104
 - управляемый уровнем синхросигнала (со статистическим управлением) 95
 - — фронтом синхросигнала (с динамическим синхронизирующим фронтом) 95, 102
 - типа D (D-триггер) 94, 101, 118, 125
 - DV (DV-триггер) 95, 101
 - JK (JK-триггер) 94, 103, 122, 126
 - RS (RS-триггер) 94, 96
 - \overline{RS} (\overline{RS} -триггер) 98
 - T (T-триггер) 95, 105, 123
- Узлы операционные (функциональные) 106
- Умножение логическое 47, 48
- Устройство арифметико-логическое (АЛУ) 116, 144
 - внешнее запоминающее 187, 207

Устройство ввода 15, 187, 214
— вывода 15
— запоминающее (ЗУ) 128
— — оперативное (ОЗУ) 128, 223
— — — динамическое 129
— — — статическое 129
— — постоянное (ПЗУ) 79, 128, 129
— — — программируемое (ППЗУ) 130
— — — репрограммируемое (РПЗУ) 130
— пересчетное (ПУ) 120
— периферийное 186
— связи с объектом 186, 188
— — с пользователем 186, 196
— печатающее 217, 218
— регистрации 216

Файл 210, 269, 270

Форма дизъюнктивная минимальная 55
— — — нормальная 55
— — — совершенная 55
— — — сокращенная 57
— записи чисел естественная 24
— — — нормальная 25
— — — с плавающей запятой 26
— — — — — со смещенным порядком 26
— — — с фиксированной запятой 25
— конъюнктивная нормальная 57
— — — совершенная 57
— — тупиковая 61, 63

Формат данных 26, 225

— команды 176, 225

Фотосчитыватель ленточный 207

Функция (операция) И 48

— ИЛИ 49
— ИЛИ-НЕ 50
— И-НЕ 49
— НЕ 47
— переключательная 46

Характеристика амплитудная передаточная (АПХ) 75, 76, 85
— входная 75
— выходная 75

Цена схемы 56, 63

Цикл выборки 147

— записи 147
— чтения 147

Шина 143, 144, 289

Шифратор 109

Элемент запоминающий (ЗЭ) 66
— импульсно - потенциальный 75

— импульсный 75
— логический (ЛЭ) 58, 66, 75, 78
— памяти 75
— потенциальный 75
— специальный 75

Эмулятор 249

Эмуляция 249

Язык высокого уровня 258

— машинно-независимый 258, 264
— машинно-ориентированный 250
— программирования 248, 291, 305

ОГЛАВЛЕНИЕ

Предисловие	3
Глава 1. Принципы организации ЭВМ	5
1.1. История развития ЭВМ	5
1.2. Принцип программного управления	12
1.3. Обобщенная структура ЭВМ	15
Контрольные вопросы и задания	17
Глава 2. Арифметические основы ЭВМ	18
2.1. Позиционные системы счисления	18
2.2. Структура и формат данных	24
2.3. Кодирование чисел	30
2.4. Сложение двоичных чисел	32
2.5. Умножение двоичных чисел	38
2.6. Деление двоичных чисел	40
Контрольные вопросы и задания	45
Глава 3. Логические основы ЭВМ	46
3.1. Двоичные переменные и переключательные функции	46
3.2. Элементарные логические функции	47
3.3. Законы булевой алгебры	52
3.4. Представление переключательных функций	54
3.5. Функционально полные системы логических функций	58
3.6. Минимизация переключательных функций	60
3.7. Основные сведения о конечном автомате	66
Контрольные вопросы и задания	73
Глава 4. Схемотехнические основы ЭВМ	74
4.1. Базовые элементы ЭВМ	74
4.2. Триггеры	93
4.3. Операционные узлы	106
4.4. Полупроводниковые БИС запоминающих устройств	128
Контрольные вопросы и задания	138
Упражнения	138
Глава 5. Организация микропроцессоров и микроЭВМ	141
5.1. Понятие об архитектуре микропроцессора и микроЭВМ	141
5.2. Однокристалльные микропроцессоры	142
5.3. Многокристалльные микропроцессоры	152
5.4. Микропрограммное управление	159
5.5. Организация памяти микроЭВМ	165
5.6. Интерфейсы микропроцессоров и микроЭВМ	167
5.7. Специальные БИС системы ввода-вывода	178
Контрольные вопросы и задания	185
Глава 6. Периферийные устройства микроЭВМ	186
6.1. Классификация периферийных устройств	186
6.2. Устройства связи с объектом	188
6.3. Устройства связи с пользователем	196
6.4. Внешние запоминающие устройства	207
6.5. Устройства ввода и регистрации	214
Контрольные вопросы	221
Упражнения	222

Глава 7. Основы программирования микроЭВМ	223
7.1. Программное обеспечение микроЭВМ	223
7.2. Система команд	225
7.3. Автоматизация программирования	247
7.4. Машинно-ориентированные языки	250
7.5. Машинно-независимые языки	258
7.6. Операционная система	268
7.7. Пакеты прикладных программ	274
7.8. Технология программирования	280
Контрольные вопросы и задания	284
Упражнения	284
 Глава 8. Применение микроЭВМ в робототехнике и системах автоматизации испытаний	 286
8.1. Организация микропроцессорных систем управления роботами	286
8.2. Система управления «Гранит-02»	292
8.3. Система управления «Сфера-36»	297
8.4. Устройство числового программного управления на базе микроЭВМ «Электроника-60»	309
8.5. Устройство числового программного управления «Электроника НЦ-31»	312
8.6. Микропроцессорная система управления АСУ-ВИ	316
8.7. Системы автоматизации термоэлектро- и климатических испытаний	331
Контрольные вопросы	337
Ответы	338
Приложения. 1. Список директив интерпретатора БЕЙСИК	343
2. Список операторов языка БЕЙСИК	343
3. Список встроенных функций транслятора БЕЙСИК	345
Литература	347
Предметный указатель	348

Учебное издание

Морозевич Анатолий Николаевич, **Дмитриев** Андрей Николаевич,
Мухаметов Валерий Николаевич, **Панькова** Валентина Владимировна,
Пашкевич Анатолий Павлович, **Путков** Виктор Никитович,
Таранов Геннадий Васильевич

МИКРОЭВМ, МИКРОПРОЦЕССОРЫ И ОСНОВЫ ПРОГРАММИРОВАНИЯ

Заведующий редакцией **А. Ф. Зиновьев**. Редактор **М. Г. Москаленко**.
Художественный редактор и художник переплета **В. Н. Валентович**.
Технический редактор **М. Н. Кислякова**. Корректор **Л. А. Шлыкович**.

ИБ № 3023

Сдано в набор 31.01.90. Подписано в печать 10.10.90. Формат 84×108/32. Бумага тип. № 1. Гарнитура литературная. Высокая печать. Усл. печ. л. 18,48. Усл. кр.-отт. 18,95. Уч.-изд. л. 19,74. Тираж 13 000 экз. Зак. 132. Цена 95 к.

Издательство «Вышэйшая школа» Государственного комитета БССР по печати. 220048, Минск, проспект Машерова, 11.

Минский ордена Трудового Красного Знамени полиграфкомбинат МППО им. Я. Коласа. 220005, Минск, ул. Красная, 23.

ПРИНЯТЫЕ

АВМ	— аналоговая вычислительная машина
АЗУ	— аналоговое запоминающее устройство
АЛУ	— арифметико-логическое устройство
АПХ	— амплитудная передаточная характеристика
АЦП	— аналого-цифровой преобразователь
БИС	— большая интегральная схема
БМУ	— блок микропрограммного управления
БС	— блок сопряжения
БФ	— блок фильтрации
ВАХ	— вольт-амперная характеристика
ВЗУ	— внешнее запоминающее устройство
ВК	— возврат каретки
ВУ	— внешнее устройство
ГПИ	— генератор прямоугольных импульсов
ГРУ	— графическое регистрирующее устройство
ГТ	— горизонтальная табуляция
ДНФ	— дизъюнктивная нормальная форма
ДРЛ	— диодно-резисторная логика
ЗУ	— запоминающее устройство
ЗЭ	— запоминающий элемент
И ² Л	— инжекционная интегральная логика
ИОН	— источник опорного напряжения
ИС	— интегральная схема
КМОП	— комплементарная МОП-логика
КНФ	— конъюнктивная нормальная форма
КОП	— код операции
КСИ	— кадровый синхронизирующий импульс
ЛИЗМОП	— лавинная инжекция заряда на МОП-транзисторе
ЛЭ	— логический элемент
МАОП	— металл-алунд-оксид-полупроводник
МГ	— магнитная головка
МДНФ	— минимальная дизъюнктивная нормальная форма
МДП	— металл-диэлектрик-полупроводник
МИС	— малая ИС
МК	— микрокоманда
МКУ	— мультиплексор кода условия
ММП	— многокристальный микропроцессор
МНОП	— металл-нитрид-оксид-полупроводник
МОП	— металл-оксид-полупроводник
МП	— микропроцессор
МПК	— микропроцессорный комплект
МПЛ	— мультиплексор
МПС	— микропроцессорная секция
МУУ	— микропрограммное устройство управления
МЭТ	— многоэмиттерный транзистор
НГМД	— накопитель на гибком магнитном диске
НКМЛ	— накопитель на кассетной магнитной ленте
НСТЛ	— транзисторная логика с непосредственными связями
ОЗУ	— оперативное ЗУ
ОЗУР	— оперативное запоминающее устройство регенерации
ОК	— открытый коллектор
ОЛ	— отрицательная логика
ОМП	— однокристальный микропроцессор
ОС	— операционная система
ОЭВМ	— однокристальная ЭВМ
ПВВ	— порт ввода-вывода
ПГС	— процессор гармонических сигналов

СОКРАЩЕНИЯ

ПДП	— прямой доступ к памяти
ПЗС	— прибор с зарядовой связью
ПЗУ	— постоянное ЗУ
ПЛ	— положительная логика
ПЛМ	— программируемая логическая матрица
ПО	— программное обеспечение
ПОС	— полный одноразрядный сумматор
ППП	— пакет прикладных программ
ППДТО	— пакет программ диагностики и технического обслуживания
ППЗУ	— программируемое ПЗУ
ПРВВ	— порт параллельного ввода-вывода
ПС	— перевод строки
ПСВВ	— порт последовательного ввода-вывода
ПСМ	— полусумматор
ПСС	— позиционная система счисления
ПСУ	— пересчетное устройство
ПУ	— периферийное устройство
ПФ	— перевод формата
Пчу	— печатающее устройство
ПЭВМ	— персональная ЭВМ
РА	— регистр адреса
РАМК	— РА микрокоманды
РД	— регистр данных
РК	— регистр команды
РКТЛ	— транзисторная логика с резисторно-конденсаторными связями
РМК	— регистр микрокоманды
РОН	— регистр общего назначения
РПЗУ	— репрограммируемое ПЗУ
РС	— регистр состояния
РСТЛ	— транзисторная логика с резисторными связями
СБИС	— сверхбольшая БИС
СДНФ	— совершенная дизъюнктивная нормальная форма
СИС	— средняя ИС
СК	— счетчик команд
СКНФ	— совершенная конъюнктивная нормальная форма
ССИ	— строчный синхронизирующий импульс
СУ	— согласующий усилитель
СУП	— схема ускоренного переноса
ТВ	— телевизионный
ТП	— токовый переключатель
УГО	— условное графическое обозначение
УР	— устройство регистрации
УСО	— устройство связи с объектом
УС	— указатель стека
УСП	— устройство связи с пользователем
УУ	— устройство управления
УФ	— ультрафиолетовое стирание
ЦАП	— цифроаналоговый преобразователь
ЦВМ	— цифровая вычислительная машина
ЦМД	— цилиндрический магнитный домен
ЦП	— центральный процессор
ША	— шина адреса
ШД	— шина данных
ЭВМ	— электронная вычислительная машина
ЭП	— эмиттерный повторитель
ЭПМ	— электрическая пишущая машинка