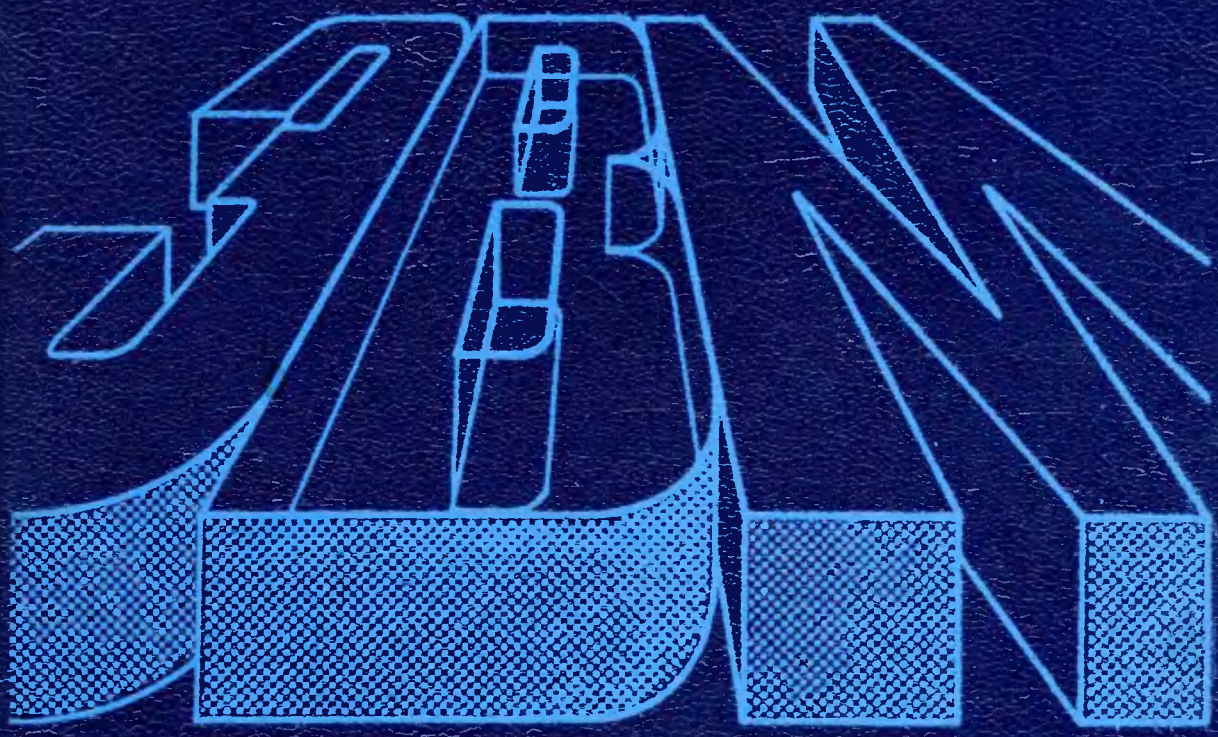
С.А.Майоров, В.В.Кириллов, А.А.Приблуда

ВВЕДЕНИЕ В МИКРО-



МАШИНОСТРОЕНИЕ

С.А.Майоров
В.В.Кириллов
А.А.Приблуда

ВВЕДЕНИЕ В МИКРОЭВМ

ЛЕНИНГРАД
„МАШИНОСТРОЕНИЕ"
ЛЕНИНГРАДСКОЕ ОТДЕЛЕНИЕ
1988

ПРЕДИСЛОВИЕ

В последнее десятилетие наблюдается чрезвычайно быстрый рост производства средств вычислительной техники, проникающей во все области человеческой деятельности — от космических исследований и производственной сферы до медицины и повседневного быта — в виде автоматизированных и автоматических систем сбора и обработки информации, управления и контроля.

Основной причиной качественно нового этапа в развитии вычислительной техники послужило появление и широкое внедрение микропроцессорных комплектов интегральных микросхем. Они позволили одновременно решить, казалось бы, несовместимые задачи: с одной стороны, резко увеличить скорость обработки информации и объем памяти, с другой — столь же резко уменьшить размеры ЭВМ, их стоимость и энергопотребление.

Вторая причина связана с тем, что средства вычислительной техники стали использовать для сбора, преобразования и хранения различной информации (справочной, результатов экспериментальных исследований, проектной документации, обучающих программ, деловых писем, статей и т. п.). Это привело к существенному расширению круга лиц, использующих ЭВМ. Количество пользователей различных информационных систем возрастает по мере развития сетей ЭВМ, позволяющих связать между собой многих поставщиков и потребителей информации, предоставляя им самые разнообразные услуги.

Наконец, дешевые микропроцессорные системы стали широко использовать для автоматизации бытовой техники и приборов, станков и игровых автоматов, роботов и автомобилей.

Все это привело к тому, что в структуре ведущих промышленных стран на первое место выдвигается индустрия ЭВМ. Эта тенденция отражается не только в постоянном росте объема вложений в соответствующие отрасли промышленности, но и в развертывании систем долговременных национальных программ создания ЭВМ новых поколений.

На совещании в ЦК КПСС (июнь 1985 г.) по вопросам ускорения научно-технического прогресса в нашей стране, а также на внеочередном заседании глав правительств стран—членов СЭВ (декабрь 1985 г.) катализаторами технического прогресса были названы: микроэлектроника, вычислительная техника и индустрия информатики. Они оказывают наибольшее влияние на эффективность средств труда и технологических систем во всех отраслях народного хозяйства.

Вследствие этого значительно повышается интерес все более широкого круга лиц самых разных профессий и возраста к микропроцессорной технике, ее потенциальным возможностям, способам встраивания микроЭВМ в конструкции машин, приборов, технологических установок, измерительных систем и другие объекты. Естественно, что одновременно растет спрос на книги, в которых бы просто и систематично были рассмотрены основы ЭВМ.

При обсуждении плана данной книги авторы долго выбирали микропроцессорный комплект интегральных микросхем, позволяющий наиболее просто и наглядно проиллюстрировать основные принципы построения и функционирования микроЭВМ. Однако каждый из промышленных микропроцессоров обладает рядом таких специфических особенностей, которые, с одной стороны, делают его наиболее удобным для использования в тех или иных приложениях, а с другой — усложняют первоначальное знакомство с микроЭВМ. Это обстоятельство, а также отсутствие веских причин, позволяющих отдать предпочтение какому-либо одному серийному микропроцессору, привело к использованию в первой части книги гипотетической микроЭВМ, которая названа базовой ЭВМ (база для ознакомления с рядом промышленных микропроцессоров и микроЭВМ). Структура базовой ЭВМ напоминает структуру очень распространенных в 60-е годы мини-ЭВМ типа «Электроника-100», а также современный микропроцессор IM 6100 [3]. Кроме того, авторы разработали модели базовой ЭВМ (см. гл. 9), использование которых позволяет работать с ней как с существующей микроЭВМ, снабженной индикаторами состояния всех ее компонентов и имеющей возможность пошагового выполнения команд и программ.

Подробное обсуждение устройства и работы базовой ЭВМ облегчило рассмотрение серийных микропроцессорных комплектов интегральных микросхем и микроЭВМ на их основе. В книге также приводятся сведения о периферийных устройствах ЭВМ, организации памяти и тенденциях развития средств вычислительной техники.

Авторы стремились изложить материал лаконично, соблюдая во всех случаях, где это возможно, практическую направленность.

Книга будет полезна практически всем, кто связан с изучением и использованием микроЭВМ, в том числе студентам и аспирантам вузов, а также школьникам при изучении основ информатики и вычислительной техники.

Предисловие и гл. 13 написаны С. А. Майоровым, гл. 1—7, 9, 12 и приложения — В. В. Кирилловым, гл. 8, 10 и 11 — А. А. Приблудой.

Авторы считают своим приятным долгом выразить искреннюю признательность канд. техн. наук А. Е. Платунову за обсуждение ряда вопросов и помощь в период подготовки рукописи.

Часть I

ОСНОВЫ ФУНКЦИОНИРОВАНИЯ И ПОСТРОЕНИЯ ЭВМ

Глава 1

ОБЩИЕ СВЕДЕНИЯ О ПРЕДСТАВЛЕНИИ И ОБРАБОТКЕ ИНФОРМАЦИИ В ЭВМ

1.1. Две формы представления информации

Как известно, современные ЭВМ могут решать самые разнообразные задачи. Для этого лишь надо с помощью программы «научить» ЭВМ алгоритму решения той или иной задачи и ввести в нее исходные данные. Программа же записывается на алгоритмическом языке (например, ФОРТРАНе или БЕЙСИКе), который достаточно близок к естественному языку (особенно английскому).

Однако ЭВМ не понимает не только естественного языка, но и алгоритмического. Для расшифровки текста программы, написанной на ФОРТРАНе, в машине должна находится специальная программа — транслятор (от англ. translate — переводить), которая переводит текст исходной программы с ФОРТРАНа на язык ЭВМ. А как же выглядит язык ЭВМ?

Вычислительная машина — это техническое устройство, в котором информация об исходных данных решаемой задачи, правилах ее решения (алгоритме) и результатах вычислений должна задаваться в виде изменения каких-либо физических величин:

углов поворота или перемещений (как, например, для «передачи информации» телевизору об уменьшении громкости или яркости);

намагниченности материала (как, например, для воспроизведения мелодии с помощью магнитофона или сохранения данных об аттестации студентов по дисциплинам учебного плана на магнитном диске);

освещенности экрана (дисплей) или фотодатчика (ввод информации с перфоленты или перфокарт).

В прошлые века, когда человечество не знало об электрических и магнитных явлениях или еще не умело их использовать, наиболее доступной, а следовательно, и удобной была механическая форма представления информации в вычислительных устройствах. В арифмометрах операции над числами выполнялись с помощью колес, которые при добавлении единицы поворачивались на 36° и с помощью штифта приводили в движение следующее по старшинству колесо всякий раз, когда цифра 9 переходила к цифре 0 (накапливался десяток). Однако механические устройства громоздки, дороги и слишком инерционны (с их помощью нельзя построить универсальные и быстродействующие вычислительные машины). Поэтому сейчас во всех вычислительных машинах в качестве основной формы представления информации служат электрические сигналы (чаще всего — напряжения постоянного

тока). Для передачи электрических сигналов нужны лишь провода, эти сигналы легко преобразовывать с помощью различных полупроводниковых схем.

При использовании в качестве носителя информации напряжений постоянного тока возможны две формы представления численного значения какой-либо переменной, например, X:

в виде одного сигнала — напряжения постоянного тока, которое сравнимо с величиной X (аналогично ей).

Например, при $X = 1845$ единиц на вход вычислительного устройства можно подать напряжение 1,845 В (масштаб представления 0,001 В/ед.) или 9,225 В (масштаб представления 0,005 В/ед.);

в виде нескольких сигналов — нескольких напряжений постоянного тока, которые, например, сравнимы с числом единиц в X, числом десятков в X, числом сотен в X и т. д.

Первая форма представления информации называется аналоговой или непрерывной (с помощью сходной величины — аналога). Величины, представленные в такой форме, могут принимать принципиально любые значения в каком-то диапазоне. Они могут быть сколь угодно близки друг к другу, малоразличимы, но все-таки, хотя бы в принципе, различимы. Количество значений, которые может принимать такая величина, бесконечно велико. Их бесконечно много даже в случае, когда величина изменяется в ограниченном диапазоне, например 0—2000 или 0—0,0001. Отсюда названия — непрерывная величина и непрерывная информация. Слово непрерывность отчетливо выделяет основное свойство таких величин — отсутствие разрывов, промежутков между значениями, которые может принимать данная аналоговая величина.

Вторая форма представления информации называется цифровой или дискретной (с помощью набора напряжений, каждое из которых соответствует одной из цифр представляемой величины). Такие величины, принимающие не все возможные, а лишь вполне определенные значения, называются дискретными (прерывистыми). В отличие от непрерывной величины количество значений дискретной величины всегда будет конечным.

Сравнивая непрерывную и дискретную формы представления информации, нетрудно заметить, что при использовании непрерывной формы создателю вычислительной машины потребуется меньшее число устройств (каждая величина представляется одним, а не несколькими сигналами), но эти устройства будут сложнее (они должны различать значительно большее число состояний сигнала). Кроме того, отметим, что устройства для обработки непрерывных сигналов обладают более высокой «квалификацией» (они могут интегрировать сигнал, выполнять любое его функциональное преобразование и т. п.) и за счет этого, а также ряда других особенностей имеют высокое быстродействие (некоторые виды задач решаются во много раз быстрее, чем с помощью устройств с дискретным представлением информации).

Однако из-за сложности технической реализации устройств для логических операций с непрерывными сигналами, длительного хранения таких сигналов, их точного измерения подобная форма представления в основном используется в аналоговых вычислительных машинах (АВМ). Эти машины предназначены для решения задач, описываемых системами дифференциальных уравнений: исследования поведения подвижных объектов (машин, роботов, судов, летательных аппаратов и т. п.); моделирования ядерных реакторов, гидротехнических сооружений, газовых сетей, электромагнитных полей и биологических систем; решения задач параметрической оптимиза-

ции и оптимального управления; управления процессами переработки нефти и выплавки стали. Но АВМ не могут решать задачи, связанные с хранением и обработкой больших объемов информации, которые легко решаются при использовании цифровой (дискретной) формы представления информации, реализуемой цифровыми электронными вычислительными машинами (ЭВМ).

Резюме

1. Исходные данные, результаты и другая информация, перерабатываемая вычислительными машинами, представляется в них в виде каких-либо физических величин — чаще всего электрических сигналов (напряжений постоянного тока).

2. Существуют две формы представления информации (физических величин) в вычислительных машинах: аналоговая (непрерывная) и цифровая (дискретная). В первой величина представляется в виде одного сигнала, пропорционального этой величине, во второй — в виде нескольких сигналов, каждый из которых соответствует одной из цифр заданной величины.

3. Непрерывная форма используется в электронных аналоговых вычислительных машинах, а дискретная — в цифровых электронных вычислительных машинах. ЭВМ — универсальные машины, позволяющие решать любые задачи науки и техники (круг таких задач все время увеличивается). Однако те задачи, где не требуется хранить и обрабатывать большие объемы информации, которые описываются системами дифференциальных уравнений и должны решаться за минимальное время, целесообразнее решать с помощью АВМ. Создают также и гибридные вычислительные системы (ГВС), которые соединяют в себе достоинства как аналоговых, так и цифровых вычислительных машин.

1.2. Способы представления дискретной информации

Как было показано выше, каждое значение из набора исходных данных задачи, результатов ее решения может быть представлено в ЭВМ в виде нескольких электрических сигналов, один из которых соответствует числу единиц в значении, другой — числу десятков, третий — числу сотен и т. д. Однако такое естественное для нас представление не является наилучшим с технических позиций. Устройство, предназначенное для обработки подобных сигналов, должно различать в каждом из них десять состояний. Значительно проще построить устройство, которое различало бы всего два состояния сигнала (его наличие или отсутствие). Это тем более целесообразно, так как существующие сейчас дешевые устройства для ввода данных в ЭВМ также кодируют (представляют) отдельные составляющие вводимой информации с помощью двух состояний:

отверстие есть или отсутствует (перфолента и перфокарта);

материал намагничен или размагничен (магнитные ленты, диски);

тумблер включен или выключен (отдельные блоки пульта управления ЭВМ);

кнопка нажата или нет (клавиатура дисплеев).

Это обстоятельство натолкнуло создателей первых ЭВМ на применение другой системы счисления при внутреннем представлении чисел в машинах: вместо привычной десятичной системы, которая использовалась в механических вычислительных устройствах, была взята двоичная система счисления. Как и десятичная система счисления, двоичная система (в которой используются лишь цифры 0 и 1) является позиционной системой счисления, т. е. в ней значение каждой цифры числа зависит от положения (позиции) этой цифры

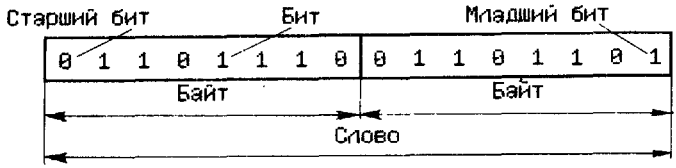


Рис. 1.1. Бит, байт и слово

в записи числа. Каждой из позиций присваивается определенный вес. Так, число 371 можно записать в виде

$$3\times10^2+7\times10^1+1\times10^0=(371),$$

$$300+70+1,$$

где цифры имеют вес 10^i, или в двоичной системе счисления

$$256+0+64+32+16+0+0+2+1=$$
$$=1\times2^8+0\times2^7+1\times2^6+1\times2^5+1\times2^4+0\times2^3+0\times2^2+1\times2^1+1\times2^0=$$
$$=(101110011)_2=(371)_{10},$$

где цифры имеют вес 2^i.

Здесь и далее основание системы счисления указано в виде индекса для числа, взятого в круглые скобки.

Существуют специальные термины, широко используемые в вычислительной технике: бит, байт и слово.

Двоичный разряд обычно называют битом. Таким образом, число 1001 является 4-битовым двоичным числом, а ранее рассмотренный двоичный эквивалент числа 371, т. е. 101110011,— 9-битовым числом. Крайний слева бит числа называется старшим разрядом (он имеет наибольший вес), крайний справа — младшим разрядом (он имеет наименьший вес). Двоичное число, состоящее из 16 бит, представлено на рис. 1.1.

Эволюция вычислительной и информационной техники вызвала появление 8-битовой единицы для обмена информацией между устройствами. Такая 8-битовая единица носит название байта. Многие новые типы ЭВМ и дискретных систем управления перерабатывают информацию порциями (словами) по 8, 16 или 32 бита (1, 2 и 4 байта). Двоичное число, состоящее из 2 байт, показано на рис. 1.1.

ЭВМ содержит большое количество ячеек памяти и регистров (от лат. regestum — внесенное, записанное) для хранения двоичной информации. Большинство этих ячеек имеет одинаковую длину n, т. е. они используются для хранения n бит двоичной информации. Информация, хранимая в такой ячейке, называется словом. Слово 16-битовой ЭВМ представлено на рис. 1.1.

Ячейки памяти и регистры состоят из элементов памяти. Каждый из таких электрических элементов может находится в одном из двух устойчивых состояний: конденсатор заряжен или разряжен, транзистор находится в проводящем или непроводящем состоянии, специальный полупроводниковый материал имеет высокое или низкое удельное сопротивление и т. п. Одно из таких физических состояний создает высокий уровень выходного напряжения элемента памяти, а другое — низкий. В элементах памяти ряда микроЭВМ это электрические напряжения порядка 4 В и 0 соответственно, причем первое обычно принимается за двоичную единицу, а второе — за двоичный нуль. (Хотя возможно и обратное кодирование.)

На рис. 1.2 показан выходной сигнал такого элемента памяти (например, одного разряда регистра) при изменении его состояний (при переключениях)

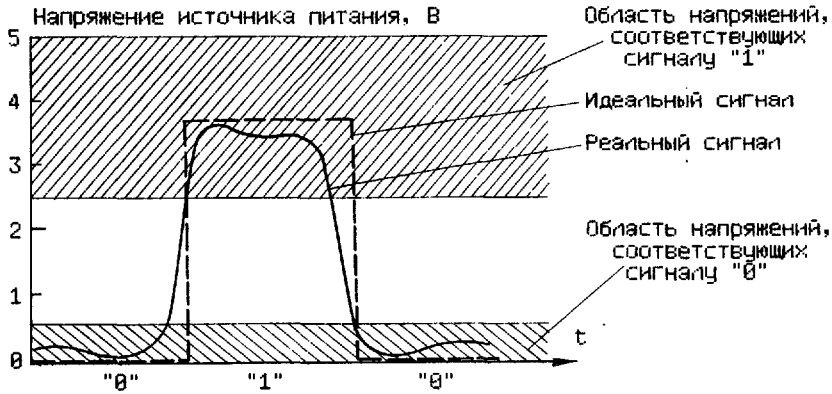


Рис. 1.2. Графическое изображение двоичного сигнала

под воздействием некоторого входного сигнала. Хотя переход от 0 к 1 и от 1 к 0 происходит не мгновенно, однако в определенные моменты времени этот сигнал достигает значений, которые воспринимаются элементами ЭВМ как 0 или 1.

Регистры часто изображают так, как показано на рис. 1.3, *а*. Регистр характеризуется единственным числом: количеством битов, которые могут в нем храниться. На рис. 1.3, *а* разряды (биты) 16-битового регистра пронумерованы степенями двойки, соответствующими позициям битов двоичного числа, находящегося в регистре. Помещенная в регистр информация остается там до тех пор, пока она не будет заменена другой. Процесс чтения информации из регистра не влияет на содержимое последнего. Говоря другими словами, операция чтения информации, хранимой в регистре, сводится к созданию копии его содержимого, оригинал же сохраняется в регистре без изменений.

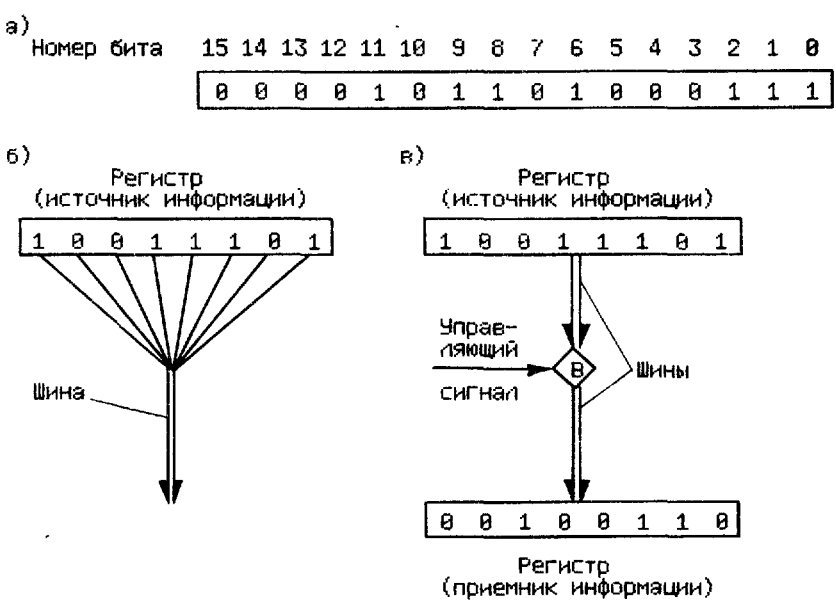


Рис. 1.3. Регистры, шины и вентильные схемы

Электрическая цепь, соединяющая регистр с другим регистром или иным устройством ЭВМ, называется шиной (bus). Шина состоит из параллельных проводов, каждый из которых предназначен для передачи соответствующего бита регистра (рис. 1.3, *б*). Два 8-битовых регистра соединяются между собой шиной из восьми проводов. Про такую шину говорят, что ее ширина равна 8. В действительности шина обычно содержит несколько дополнительных проводов, используемых для передачи сигналов синхронизации и управления, однако подобный анализ структуры шин нас пока не интересует.

Следует отметить лишь одно. Несмотря на то что электрический провод позволяет передавать сигналы в любом направлении, шина служит для передачи информации лишь в направлении, обозначенном стрелкой на шине (рис. 1.3, *б, в*). Такая однонаправленность передачи обусловлена не свойствами шины, а характеристиками схем, соединяющих шину с регистром или другими устройствами ЭВМ. Специальные схемы позволяют, например, в одни моменты времени передавать информацию по шине в одну сторону, а в другие — в обратном направлении, т. е. организовать двунаправленную шину.

Вентильные схемы — это электронные ключевые схемы, предназначенные для управления потоком информации из регистров в шины и обратно. На рис. 1.3, *в* показано применение вентильной схемы *В*. Такая схема имеет два входа и один выход. На один вход подается информационный сигнал (данные с регистра), а на другой — управляющий. Если управляющий сигнал равен единице, то данные проходят через схему без препятствий, как будто ее и нет. Если управляющий сигнал равен нулю, никакая информация не пройдет через схему. Для подачи информационного сигнала на вход вентильной схемы обычно используется многопроводная шина. Для передачи выходного сигнала вентильной схемы требуется шина с таким же количеством проводов. Если управляющий сигнал равен единице, то информационные сигналы на входной и выходной шинах совпадают. Таким образом, при подаче единичного управляющего сигнала на вентильную схему *В* состояние регистра приемника информации изменится с $(00100110)_2$ на $(10011101)_2$, т. е. в него поступит копия содержимого регистра источника информации.

Тактовые импульсы (вырабатываемые генератором тактовых импульсов ЭВМ) используются для синхронизации процессов передачи информации между устройствами ЭВМ. Поскольку на простом примере трудно проиллюстрировать необходимость распределения во времени отдельных операций по передаче информации в цифровых устройствах, потратим немного времени на рассмотрение работы упрощенной модели школьного микрокалькулятора (рис. 1.4, *а*).

В состав микрокалькулятора входят:

цифровая (0, 1, ..., 9, «.») и функциональная (C, %, ×, ÷, −, √, +, =) клавиатуры;

основной регистр *X*, в который вводится операнд (набираемое на клавиатуре число) и по окончании какой-либо операции засылается ее результат (сумма, разность и т. п.);

табло для индикации содержимого регистра *X*, т. е. вводимого числа (в момент набора его на цифровой клавиатуре) или результата вычислений (после нажатия функциональной клавиши);

вспомогательный регистр *Y*, в который переписывается содержимое регистра *X* при вводе в последний нового операнда (это позволяет, например, сохранить в *Y* значение предыдущего слагаемого или ранее накопленной суммы в момент ввода в *X* нового слагаемого);

арифметико-логическое устройство (АЛУ), предназначенное для выполнения операций над содержимым регистров *X* и *Y* (сложение, умножение,

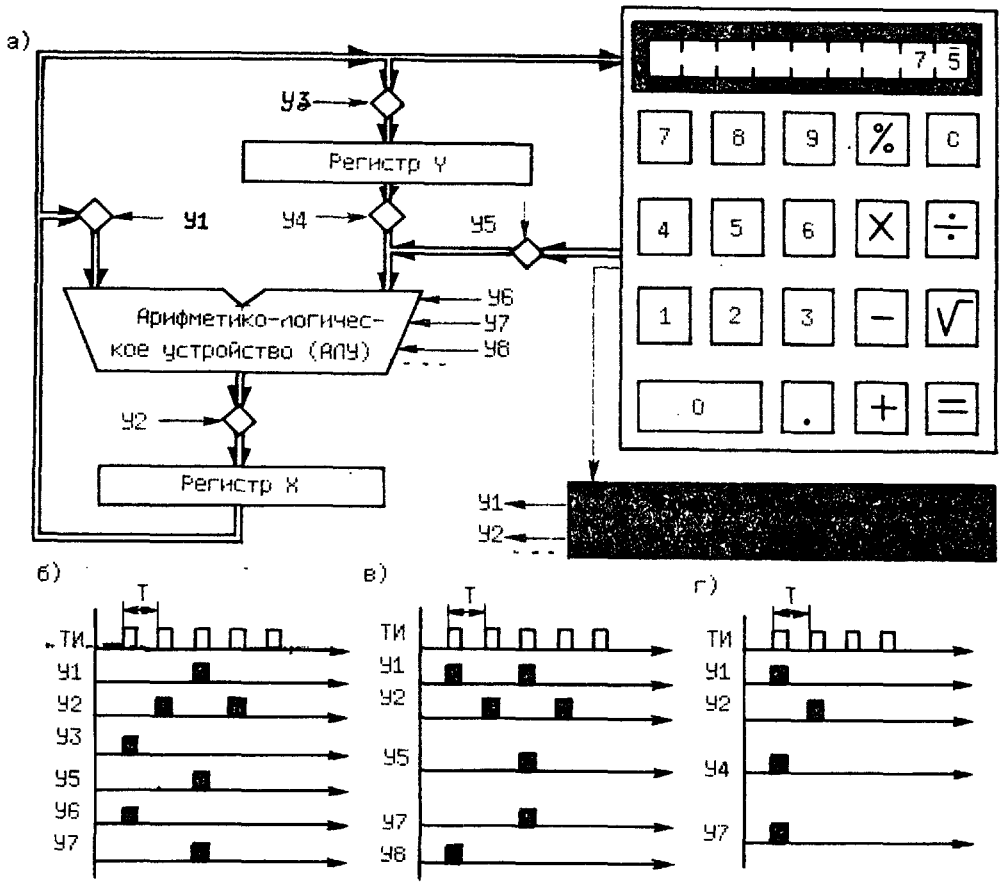


Рис. 1.4. К необходимости синхронизации процессов передачи информации между устройствами ЭВМ

вычитание и деление) или только регистра X (извлечение квадратного корня и нахождение процентов), а также ряда служебных операций (например, сдвига старших цифр числа при вводе его новой цифры);

устройство управления вводом числа и выполнением операций, которое после любого нажатия цифровой или функциональной клавиши вырабатывает определенную последовательность управляющих сигналов ($У_i$) для осуществления необходимых пересылок информации между устройствами микрокалькулятора и (или) перестройки АЛУ на ту или иную операцию.

Кроме того, отметим, что хотя микрокалькулятор работает с сигналами, которые кодируются двоичными цифрами 0 и 1, в примере будет использовано знакомое десятичное представление цифр в регистрах. Это оправдывается еще и тем, что числа, обрабатываемые в микрокалькуляторах, кодируются в так называемой двоично-десятичной системе счисления.

Что будет происходить с регистрами микрокалькулятора в процессе ввода слагаемых и получения суммы чисел $27+48+12976+38+\ldots$? Начнем с момента нажатия клавиши «+», завершающей ввод первого слагаемого, т. е. с момента, когда в регистрах содержатся следующие числа:

Регистр Y	0
Регистр X	27

1-й шаг. Ввод цифры 4 — старшей цифры второго слагаемого:

Регистр Y	27
Регистр X	4

Содержимое регистра X переписалось для сохранения в регистр Y, регистр X был очищен и в его младший разряд записан код нажатой клавиши, т. е. цифра 4.

2-й шаг. Ввод цифры 8 — младшей цифры второго слагаемого:

Регистр Y	27
Регистр X	48

Число, содержащееся в регистре X, было сдвинуто влево на один разряд, а в освободившийся младший разряд записан код нажатой клавиши, т. е. цифра 8.

3-й шаг. Нажатие клавиши «+»:

Регистр Y	27
Регистр X	75

Зафиксировался ввод последней цифры второго слагаемого (48), выполнились операция сложения содержимого регистров X и Y ($48+27=75$) и операция пересылки полученного результата в регистр X.

4-й шаг. Ввод цифры 1 — старшей цифры третьего слагаемого:

Регистр Y	75
Регистр X	1

Содержимое регистра X переписалось... (см. комментарии к 1-му шагу).

Дальнейшие шаги не представляют интереса, так как в них будут повторяться ранее описанные операции. Поэтому перейдем к рассмотрению последовательности управляющих сигналов $У_i$, вырабатываемых устройством управления при выполнении трех первых шагов.

Ввод первой цифры любого операнда (первое нажатие цифровой клавиши после выполнения какой-либо операции) должен инициировать перепись содержимого регистра X в регистр Y, т. е. выработку управляющего сигнала *У3* (рис. 1.4, *б*). Одновременно с этим сигналом может быть выработан сигнал *У6*, устанавливающий 0 на выходе АЛУ. Теперь производятся очистка регистра X, суммирование его содержимого с вводимой цифрой и перепись полученной суммы в регистр X. Для этого сначала подается сигнал *У2* (0 с выхода АЛУ переписывается в регистр X), затем одновременно сигналы *У7, У1* и *У5* (*У7* для настройки АЛУ на суммирование) и, наконец, сигнал *У2*.

Ввод второй (или любой из следующих цифр) осуществляется путем умножения на 10 (сдвига влево на один десятичный разряд) содержимого регистра X и суммирования сдвинутого значения с вводимой цифрой. Для этого вырабатывается последовательность управляющих сигналов, показанная на рис. 1.4, *в* (сигнал *У8* служит для организации сдвига).

Суммирование содержимого регистров X и Y (сигналы *У7, У1* и *У4*) и перепись полученной суммы в регистр X (сигнал *У2*) иллюстрируются временной диаграммой на рис. 1.4, *г*.

Обсудим, наконец, почему некоторые из управляющих сигналов временны́х диаграмм на рис. 1.4, *б, в, г* вырабатываются одновременно, а другие — через те или иные интервалы времени. Например, можно ли при суммировании содержимых регистров X и Y одновременно вырабатывать сигналы *У1, У2, У4*

У7? Простейший анализ позволяет установить, что при одновременной выработке сигналов *У1* и *У2*, а также при настройке АЛУ на выполнение какой-либо операции с содержимым регистра *X* возникает полная неопределенность: содержимое регистра *X* попадает в АЛУ, через него обратно в регистр *X* и т. д. С другой стороны, если при суммировании *X* и *Y* не будет обеспечена одновременная выработка сигналов *У1*, *У4* и *У7*, то суммы не получится.

А чем же определяется минимальный промежуток времени между управляющими сигналами, которые должны быть сдвинуты относительно друг друга (например, промежуток времени между сигналами *У1* и *У2* на рис. 1.4, *г*)? Этот промежуток времени *T* выбирается исходя из максимального времени переключения элементов ЭВМ (пока, например, не окончилось формирование суммы $X+Y$, бессмысленно подавать управляющий сигнал *У2* на перепись этой суммы в регистр *X*). Для выдерживания нужных промежутков времени в управляющих устройствах ЭВМ устанавливаются генераторы тактовых импульсов (ГТИ), синхронизирующие управляющие сигналы $У_i$.

Из приведенных выше примеров видно, что с помощью электрических сигналов можно достаточно просто представлять двоичные цифры. Эти сигналы (цифры) легко пересылать между устройствами ЭВМ, они могут изменять состояния устройств для хранения цифр (элементов памяти регистров и ячеек памяти) и т. д. Однако во всех примерах рассматривались лишь двоичные эквиваленты целых десятичных чисел без знака и могло сложиться впечатление, что 0 и 1 нельзя использовать для кодирования и обработки другой информации. Это не так, наборы двоичных цифр (так же, как наборы точек и тире в азбуке Морзе) позволяют закодировать любую информацию (вплоть до графической и звуковой).

Если надо закодировать целое число со знаком, то старший бит регистра (ячейки памяти) используется для хранения знака (например, 0 при положительном знаке числа и 1 при отрицательном). На рис. 1.5, *а* показано такое представление числа со знаком для 16-битовой ЭВМ. Числа вида $\pm M \cdot 10^{\pm p}$

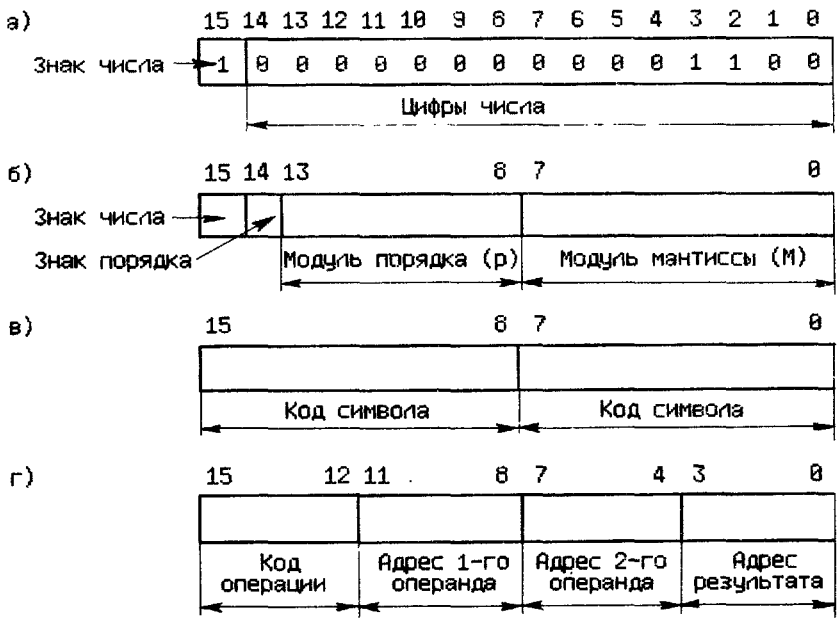


Рис. 1.5. Форматы представления информации в 16-битовой ЭВМ:
а — целые числа; *б* — числа с плавающей запятой; *в* — символы; *г* — команды

(например, $384 \cdot 10^{-3}$ или $-9 \cdot 10^{-5}$) могут кодироваться так, как показано на рис. 1.5, *б*. Однако в 16 разрядах нельзя сохранить хоть сколько-нибудь приемлемое число цифр мантиссы и чаще всего для кодирования подобных чисел используются слова длиной от 32 до 64 бит. Буквы русского и латинского алфавитов, знаки операций и другие символы кодируются с помощью 8-битовых (1-байтовых) чисел (рис. 1.5, *в*), что позволяет закодировать $2^8 = 256$ различных символов (см. табл. Б.1 и Б.2 приложения Б).

Аналогичным образом (в виде двоичных чисел) кодируются команды — приказы на выполнение каких-либо операций со словами информации. Так, в четырех старших битах команды на рис. 1.5, *г* закодирована операция (например, вычитание), далее адреса ячеек памяти, из которых надо взять операнды (уменьшаемое и вычитаемое), и, наконец, адрес ячейки памяти, в которую помещается результат выполненной операции (разность).

Когда нам приходится сталкиваться с информацией вида 50 369, —15 167, «Да» или «C4C1; перейти к команде, расположенной в ячейке памяти с адресом $(4C1)_{16}$», мы легко различаем перечисленные объекты по их внешнему виду: на первом месте расположено целое число без знака, на втором — целое число со знаком, на третьем — русское слово «Да» (так как использован символ русского алфавита — «Д»), а на четвертом — команда безусловного перехода базовой ЭВМ с пояснением к ней.

Однако без пояснений нельзя понять, какую информацию несет набор битов 1100 0100 1100 0001, с помощью которого кодируются в ЭВМ все перечисленные выше объекты*. А как же различает их ЭВМ? Есть два пути, позволяющие информировать ЭВМ о том, что закодировано в принимаемой ею последовательности битов.

Первый путь связан с дополнением последовательности рядом идентифицирующих битов — метабитов. Так, вводя в последовательность один дополнительный метабит и записывая в него, например, 0 для команд и 1 для других видов информации, можно легко научить ЭВМ отличать команды от данных. Несколько метабитов помогли бы однозначно определять символы, целые числа и т. д. Однако при таком способе идентификации последовательности битов требуется увеличивать объем памяти ЭВМ для хранения метабитов, усложнять ее структуру за счет введения схем анализа метабитов, т. е. необходимы дополнительные затраты как материальных ресурсов, так и времени (на анализ метабитов).

Второй путь, используемый при работе с большинством современных ЭВМ, не связан с затратой материальных или временных ресурсов, но от программиста требуются детальное распределение в памяти ЭВМ команд программы и данных, точное указание в командах адресов операндов (результата или перехода) и осуществление ряда других мероприятий, исключающих возможность применения операнда в качестве команды, команды — в качестве операнда и внесения прочих ошибок, которые не могут быть выявлены в процессе выполнения программы. Труд программиста несколько облегчается при использовании им для написания программ какого-либо алгоритмического языка, но и в этом случае приходиться детально описывать типы переменных, размерности массивов и т. п.

Резюме

1. Конструкция ЭВМ предельно упрощается и ЭВМ работает наиболее надежно (устойчиво), если сигналы, циркулирующие в электронных схемах ЭВМ, используются для представления только двух значений: 0 и 1.

---

* Число —15 167 закодировано в так называемом дополнительном коде, о котором рассказано в параграфе 2.4.

2. Бит (от англ. binary digit) — двоичная цифра. Бит может принимать лишь одно из двух значений — 0 или 1 (да или нет, включено или выключено).

3. Группа из восьми соседних разрядов (битов), с которой ЭВМ оперирует как с одним целым при передаче, хранении и обработке информации, называется байтом. Байт обычно используется для представления либо букв и специальных символов, либо пары десятичных или шестнадцатеричных цифр.

Более крупные единицы информации — слова. Они обычно кратны по длине байту, и это значительно упрощает согласование процессов обработки информации в ЭВМ.

4. Двоичные цифры представляются в ЭВМ сигналами (чаще всего напряжениями), четко различаемыми элементами машины. Хотя эти сигналы непрерывны, смысл двоичных символов (0 или 1) они имеют в определенные дискретные моменты времени, которые чаще всего задаются с помощью тактирующих импульсов.

5. С помощью последовательности двоичных цифр можно закодировать, хранить и обрабатывать в ЭВМ любую информацию (числа, тексты, команды и т. п.). Разнотипные слова информации обычно отличаются друг от друга способом использования, но не способами кодирования.

1.3. Системы счисления, используемые в вычислительной технике

Удобная для использования в ЭВМ двоичная система счисления совсем неудобна для записи и чтения чисел человеком. Действительно, вместо, например, четырехзначного десятичного числа $(8769)_{10}$ приходится работать с его 14-разрядным эквивалентом — двоичным числом $(10001001000001)_2$. Правда, к этому прибегают лишь при общении с ЭВМ на уровне ее машинного языка. Однако в настоящее время такой язык часто используется при работе с микропроцессорами и микроЭВМ.

Для сокращения трудоемкости ручной обработки кодов чисел, команд широко применяют восьмеричную и шестнадцатеричную системы счисления. В восьмеричной системе счисления используются 8 цифр (0, 1, 2, 3, 4, 5, 6, 7), в шестнадцатеричной — 10 цифр и 6 прописных латинских букв от A до F (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F). Так как основанием восьмеричной системы является число $8=2^3$, то для перевода двоичных чисел в восьмеричные необходимо разделить двоичные числа на 3-битовые группы — триады. Каждую из таких групп можно представить одной восьмеричной цифрой, используя табл. А.1 эквивалентности кодов (см. приложение А). Аналогичным образом осуществляется перевод двоичных чисел в шестнадцатеричные $(16=2^4)$. Только в этом случае двоичное число разбивается на 4-битовые группы (тетрады), которые и представляются одной шестнадцатеричной цифрой. Покажем два примера подобных преобразований:

шестнадцатеричное		3	5	0	6
двоичное	0	0 1 1	0 1 0	1 0 0	0 0 0 1 1 0
восьмеричное		3	2	4	0 6

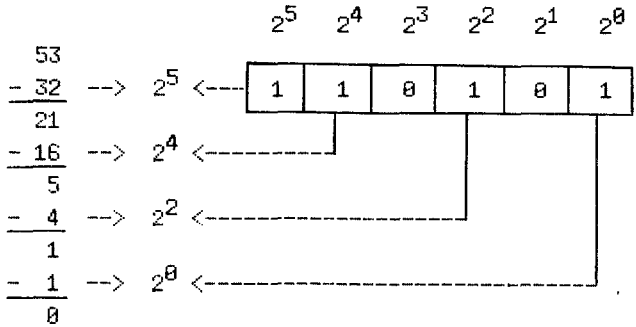
шестнадцатеричное		B	E	A	C
двоичное	1 0 1 1	1 1 1 0	1 0 1 0	1 1 0 0	
восьмеричное	1 3	7	2	5	4

Преобразование двоичных чисел в десятичные может быть выполнено различными способами. Простейший из них вытекает непосредственно из определения двоичных чисел. Преобразование осуществляется путем суммирования значений степеней числа 2, соответствующих тем разрядам переводимого двоичного числа, в которых содержатся единицы. Например:

7	6	5	4	3	2	1	0	разряды переводимого числа
1	0	1	1	1	0	1	0	переводимое число
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	степень числа 2
128	+0	+32	+16	+8	+0	+2	+0	=186 результат перевода

Следовательно, $(10111010)_2 = (186)_{10}$.

Обратное преобразование можно выполнить, в частности, следующим образом. Из переводимого десятичного числа вычитается наибольшее значение степени числа 2, не превышающее заданного числа. Затем операция повторяется для полученной разности (остатка). Как только заданное число окажется полностью разложенным на значения степеней 2, его искомое двоичное выражение можно скомпоновать из единиц в битовых позициях, соответствующих имеющимся в разложении степеням 2, и нулей во всех прочих позициях. Например:



Следовательно, $(53)_{10} = (110101)_2$.

Значения степеней числа 2, которые используются для указанных переводов, приведены в табл. А.2 приложения А. Первая строка этой таблицы позволяет получить степени числа 16 (16^0, 16^1, 16^2, 16^3, 16^4).

Аналогичным образом можно выполнить перевод восьмеричных и шестнадцатеричных чисел в десятичные и наоборот. При переводе двоичных и восьмеричных чисел в шестнадцатеричные рекомендуем использовать таблицы приложения А.

Арифметические операции с двоичными, восьмеричными и шестнадцатеричными числами осуществляются по тем же правилам, что и с десятичными числами, за исключением того, что переносы в следующие разряды производятся при достижении 2, 8 и 16, а не 10 как в десятичной системе. Подробно ряд арифметических операций рассмотрим в параграфе 2.4, а здесь приведем пример сложения чисел $(166)_{10}$ и $(47)_{10}$ в двоичной, восьмеричной и шестнадцатеричной системах счисления:

Перенос	1 111	11	1
1-е слагаемое	10100110	246	A6
2-е слагаемое	00101111	57	2F
Результат	11010101	325	D5

Наконец, следует упомянуть и о двоично-десятичной системе счисления, которая широко используется в цифровых устройствах, где основная часть

операций связана не с обработкой и хранением вводимой информации, а с самим ее вводом и выводом на какие-либо индикаторы с десятичным представлением полученных результатов (микрокалькуляторы, кассовые аппараты и т. п.).

В двоично-десятичной системе десятичные цифры от 0 до 9 представляют 4-разрядными двоичными комбинациями от 0000 до 1001, т. е. двоичными эквивалентами десяти первых шестнадцатеричных цифр (см. табл. А.1 приложения А). Преобразования из двоично-десятичной системы в десятичную (и обратные преобразования) не вызывают затруднений и выполняются путем прямой замены четырех двоичных цифр одной десятичной цифрой (или обратной замены). Две двоично-десятичные цифры составляют 1 байт. Таким образом, с помощью 1 байта можно представлять значения от 0 до 99, а не от 0 до 255 или от 0 до FF, как при использовании 8-разрядного двоичного числа или 2-разрядного шестнадцатеричного числа. Используя 1 байт для представления каждых двух десятичных цифр, можно формировать двоично-десятичные числа с любым требуемым числом десятичных разрядов.

Так, если число

$$1001\ 0101\ 0011\ 1000$$

рассматривать как двоичное, то его десятичный эквивалент

$$(1001\ 0101\ 0011\ 1000)_2 = (38200)_{10}$$

в 4 раза больше десятичного эквивалента двоично-десятичного числа

$$(1001\ 0101\ 0011\ 1000)_{2-10} = (9538)_{10}.$$

Сложение двоично-десятичных чисел, имеющих один десятичный разряд, выполняется так же, как и сложение 4-разрядных двоичных чисел без знака, за исключением того, что при получении результата, превышающего 1001, необходимо производить коррекцию. Результат корректируется путем прибавления двоичного кода числа 6, т. е. кода 0110. Например:

```
  4        0100        5        0101                9        1001
+ 5      + 0101      + 9      +  1001              + 9      +  1001
-----    --------    ----     --------             ----     --------
  9        1001       14       1110                18       1 0010
                              + 0110  — коррекция          + 0110  — коррекция
                   10+4       --------             10+8     --------
                              1 0100                       1 1000
```

Если первоначальное двоичное сложение или прибавление корректирующего числа приводит к возникновению переноса, то при сложении многоразрядных двоично-десятичных чисел осуществляется перенос в следующий десятичный разряд:

```
  1889      0001      1000      1000      1001
+ 6376    + 0110    + 0011    + 0111    + 0110
-------   -------   -------   -------   -------
  8265      0111      1011      1111      1111
                    + 0110    + 0110    + 0110
                              + 1       -------
            +   1   -------   -------     0101
          -------     0010      0110
            1000
```

Резюме

1. Для удобства выполнения ручных операций по подготовке и отладке программ для микроЭВМ двоичные коды команд и данных обычно заменяют на восьмеричные или шестнадцатеричные.

2. Перевод двоичных чисел в восьмеричные (шестнадцатеричные) выполняется путем разбиения этого числа на 3-битовые (4-битовые) группы и замены каждой из групп на восьмеричную (шестнадцатеричную) цифру.

3. Перевод восьмеричных (шестнадцатеричных) чисел в двоичные осуществляется путем замены каждой восьмеричной (шестнадцатеричной) цифры на ее 3-битовый (4-битовый) двоичный код.

4. Двоичное число можно преобразовать в десятичное, представив его в виде суммы степеней числа 2.

5. Десятичное число можно преобразовать в двоичное путем последовательного вычитания степеней числа 2, начиная с наибольшей степени, содержащейся в числе. Найденные в результате коэффициенты располагают в последовательности таким образом, чтобы коэффициент при старшей степени числа 2 стоял в старшем разряде двоичного числа.

6. Ряд микроЭВМ может обрабатывать информацию, представленную в виде двоичных кодов десятичных цифр, т. е. как бы в десятичной системе счисления. Подобная система счисления получила название двоично-десятичной. Перевод чисел из десятичной системы счисления в двоично-десятичную (и обратное преобразование) выполняется путем прямой замены каждой десятичной цифры числа на четыре двоичные (или обратной замены).

7. Основание системы счисления часто указывают в виде индекса для числа, взятого в круглые скобки.

1.4. Структура и принцип функционирования ЭВМ

Типичная ЭВМ состоит из процессора, памяти и устройств ввода-вывода (рис. 1.6). Со времени появления в 40-х годах первых электронных цифровых вычислительных машин технология производства каждой из этих трех подсистем была значительно усовершенствована. За последнее десятилетие благодаря развитию интегральной технологии существенно улучшились характеристики процессоров и памяти. Кроме того, была снижена их стоимость. В настоящее время по цене хорошего цветного телевизора можно приобрести в личное пользование достаточно мощную ЭВМ, за которую в 70-х годах потребовалось бы заплатить больше, чем за несколько самых дорогих автомобилей.

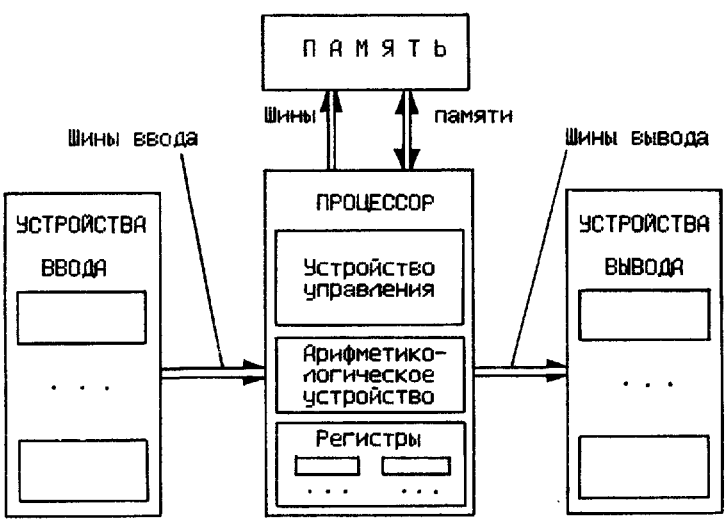


Рис. 1.6. Обобщенная структурная схема ЭВМ

Несмотря на успехи, достигнутые в области технологии, существенных изменений в базовой структуре и принципах работы вычислительных машин не произошло. Так, еще в 1946 г. в описании впервые предложенной ЭВМ с хранимой в памяти программой отмечалось: «Мы располагаем... двумя различными типами памяти: памятью чисел и памятью команд. Тем не менее если команды представлены машине в виде числового кода и если машина каким-либо образом в состоянии отличать числа от команд, то блок памяти можно использовать для хранения и тех и других».

И в настоящее время почти во всех вычислительных машинах для хранения данных (чисел, текстов) и команд служит одна и та же память. Это позволяет повысить эффективность использования достаточно дорогостоящей памяти ЭВМ, так как среди решаемых ею задач встречаются задачи с достаточно сложной обработкой (много команд) небольшого числа исходных данных (например, ряд научных расчетов) и задачи, связанные с переработкой по простым алгоритмам (мало команд) очень больших объемов данных (обработка переписи населения, расчет заработной платы на крупном предприятии и т. п.).

«Сердцем» ЭВМ является *процессор*, в состав которого входят:

устройство управления выборкой команд из памяти и их выполнением;

арифметико-логическое устройство, производящее операции над данными;

регистры, осуществляющие временное хранение данных и состояний процессора;

схемы для управления и связи с подсистемами памяти и ввода-вывода.

Подробнее о структуре процессора будет рассказано в конце этого параграфа.

Устройства ввода обеспечивают считывание информации (исходных данных и программы решения задачи) с определенных носителей информации (клавиатур, перфолент, магнитных лент или дисков, датчиков состояний управляемых объектов и т. п.) и ее представление в форме электрических сигналов, воспринимаемых другими устройствами ЭВМ (процессором или памятью).

Устройства вывода представляют результаты обработки информации в форме, удобной для визуального восприятия (индикаторы, печатающие устройства, графопостроители, экран дисплея и т. п.). При необходимости они обеспечивают запоминание результатов на носителях, с которых эти результаты могут быть снова введены в ЭВМ для дальнейшей обработки (перфоленты, магнитная лента, магнитный диск и т. п.), или передачу результатов на исполнительные органы управляемого объекта (например, робота).

Память ЭВМ включает устройство, обеспечивающее хранение команд и данных. Это устройство состоит из блоков одинакового размера — *ячеек памяти*, предназначенных для хранения одного слова информации (рис. 1.7, *а*). В свою очередь, ячейка памяти состоит из элементов памяти, состояние каждого из которых соответствует одной двоичной цифре (0 или 1). Совокупность нулей и единиц, хранящихся в элементах одной ячейки, представляет собой содержимое этой ячейки памяти.

Ячейки нумеруются числами 0, 1, 2, ..., называемыми *адресом* ячеек. Если необходимо записать в память слово, следует подать на шину адреса памяти (рис. 1.7, *б*) сигналы, соответствующие адресу ячейки, в которую надо поместить записываемое слово, и подать само слово на шину записи. Память устроена так, что заданное слово будет передано в ячейку с указанным адресом и может храниться там сколь угодно долго. В любой момент, обратившись к памяти, можно получить содержимое хранимого там слова. Для этого в память посылается адрес, определяющий местоположение

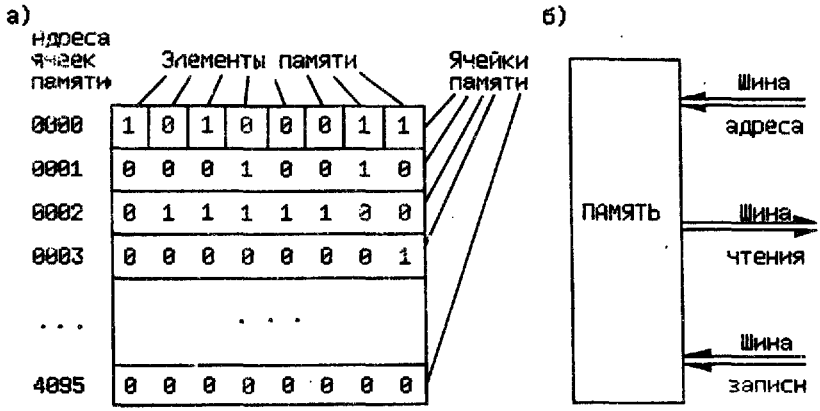


Рис. 1.7. Организация памяти ЭВМ

требуемого слова, и она выдает по шине чтения копию слова. При считывании содержимое ячейки остается без изменения, так что, один раз записав слово, можно сколько угодно раз получать его копии. Это аналогично записи песни на магнитофонную ленту. Песню можно прослушивать с ленты (читать с ленты) сколько угодно раз, но если на ее место записать другую мелодию, то первая будет стерта. Однако время доступа к информации на магнитной ленте зависит от того, где записана эта информация (иногда надо перемотать почти всю ленту, чтобы прослушать какую-либо песню), тогда как время доступа к любой ячейке памяти всегда одинаково (не зависит от ее номера).

На рис. 1.7, *а* показана память, имеющая $4096 = 2^{12}$ 8-разрядных слов, т. е. содержащая 4096 байт информации. При том же самом числе запоминающих элементов можно было бы организовать память из 32 768 1-битовых слов, 2048 16-битовых слов, 1024 32-битовых слов и т. д. Если нужно обрабатывать информацию, которая может кодироваться лишь двумя символами (например, некоторые данные переписи населения: пол, семейное положение и т. п.), то выгодно использовать первую организацию памяти. В случае же точных вычислений приемлемее память из 32-битовых или даже 64-битовых слов. Однако при выборе разрядности ячеек памяти надо учитывать, что в них должны храниться и команды программы, используемые процессором для обработки таких данных. А как же строится команда ЭВМ и какова ее разрядность?

Команда ЭВМ первоначально содержала следующую информацию (рис. 1.8, *а*).

1. Код операции, указывающий операцию, которую должна выполнить ЭВМ (сложение, вычитание, умножение, сравнение, изменение знака и т. п.).

2. Адреса двух операндов — аргументов операции, например слагаемых, уменьшаемого и вычитаемого, сомножителей и т. п. Если какой-либо из операндов является константой, то вместо его адреса в команде может быть задано само значение операнда. Однако это обстоятельство должно быть отражено в коде операции, чтобы ЭВМ использовала соответствующую часть команды в качестве операнда, а не адреса ячейки памяти, в которой хранится этот операнд.

3. Адрес ячейки памяти, в которую должен быть помещен результат операции.

4. Адрес следующей команды.

Такая команда, состоящая из пяти полей (код операции и четыре адреса), может быть реализована процессорами самых разнообразных структур.

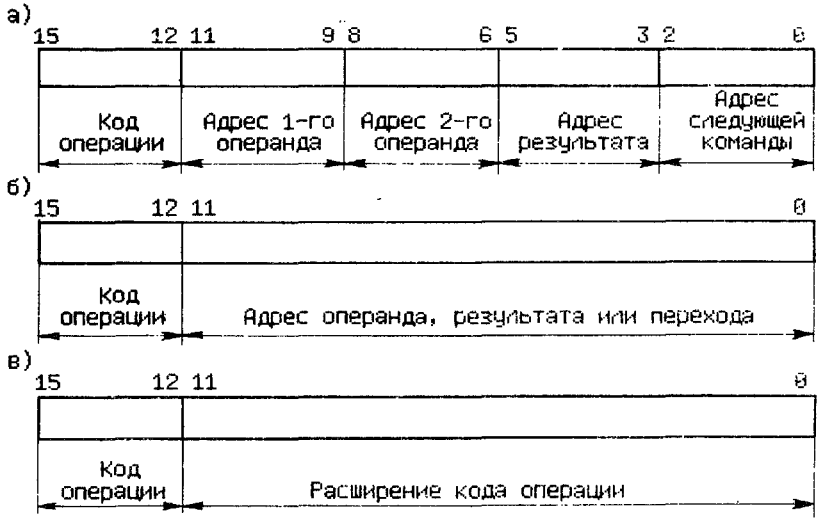


Рис. 1.8. Форматы команд:

а — четырехадресная; *б* — одноадресная; *в* — безадресная

Простейшая из этих структур (рис. 1.9) напоминает структуру процессора микрокалькулятора, приведенную на рис. 1.4. Процессор содержит устройство управления, АЛУ, регистр для размещения исполняемой команды (регистр команд) и регистр для размещения одного из операндов или результата операции в процессе выполнения этой команды — аккумулятор (на рис. 1.4, *а* роль аккумулятора выполнял регистр *X*). Рассмотрим, что происходит в процессоре после того, как в его регистр команд была переписана из памяти какая-либо команда, например команда вычитания.

Так как поле регистра команд, в котором хранится код операции, связано шиной с устройством управления, то последнее получит приказ на выполнение операции «Вычитание» и перейдет в режим генерации управляющих сигналов:

одновременной выработкой сигналов *У1, У5, У10* и *У8* обеспечиваются пересылка из памяти численного значения уменьшаемого, прохождение его через АЛУ (сигнал *У10*) и запись в аккумулятор;

следующий тактовый импульс инициирует выработку сигналов *У2, У5, У6* и *У11*, что приводит к пересылке из памяти численного значения вычитаемого, к выполнению операции вычитания этого значения (сигнал *У11*) из содержимого аккумулятора (из уменьшаемого); по сигналу *У8* производится запись полученной разности в аккумулятор вместо уменьшаемого;

затем сигналы *У3* и *У9* обеспечат пересылку разности из аккумулятора в ячейку памяти, на которую указывает предпоследнее поле команды;

наконец, с помощью сигналов *У4* и *У7* будет произведена перепись в регистр команд следующей команды программы.

Рассмотренная структура процессора и реализуемая им структура четырехадресной команды кажутся вполне естественными. Действительно, без усложнения доступа к содержимому ячеек памяти нельзя одновременно получать из нее оба операнда и, следовательно, первый операнд приходится сохранять в процессоре до момента получения следующего операнда. Поэтому в состав процессора и включен аккумулятор (он назван так потому, что может быть использован для накопления — аккумуляции — слагаемых в процессе получения суммы многих слагаемых или накопления результатов других вычислений).

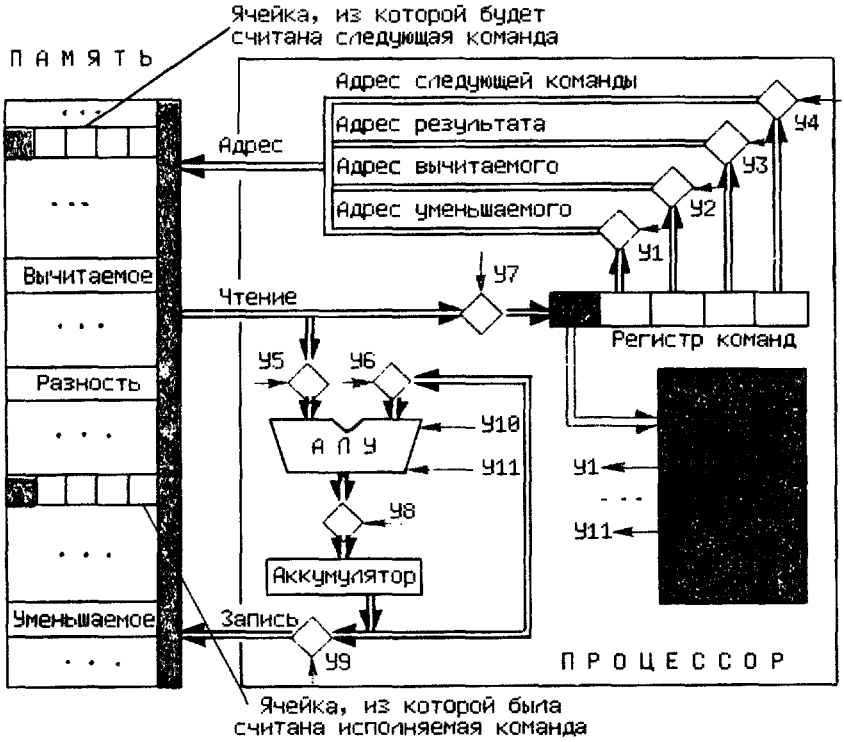


Рис. 1.9. Простой процессор, работающий с четырехадресной командой, целиком выбираемой из одной ячейки памяти

Нужен и регистр для хранения в процессоре исполняемой команды (регистр команд), так как во время выполнения этой команды из нее должна выбираться различная информация, используемая устройством управления и памятью. Однако целесообразно ли хранить в регистре команд одновременно все поля команды, если они все равно используются последовательно?

Когда команда хранится в одной ячейке памяти, то не существует возможности считывания из памяти отдельных полей этой команды. Поэтому лучше поставить вопрос так: целесообразно ли хранить четырехадресную команду в одной ячейке памяти или лучше хранить ее отдельные поля в нескольких ячейках памяти?

Короткую четырехадресную команду (см. рис. 1.8, *а*) безусловно следует целиком хранить в 16-битовой ячейке памяти. Но такая команда может адресовать лишь $2^3=8$ ячеек памяти. Современные же микроЭВМ адресуются к памяти, содержащей не менее $2^{16}=65\,536$ ячеек. Если создавать четырехадресную команду, работающую с памятью подобных размеров, то на каждое адресное поле команды придется отвести 16 бит, а на всю команду (при 16-битовом коде операции) — 80 бит. Однако ЭВМ, работающую с такими командами, уже нельзя назвать микроЭВМ, так как она будет содержать очень много оборудования: 80-разрядные ячейки памяти, 80-разрядные регистры, 80-разрядное АЛУ, 80-разрядные вентильные схемы, шины шириной 80 и т. д. Следовательно, надо либо четырехадресную команду хранить в ячейках памяти в виде отдельных ее полей, либо уменьшать число адресных полей команды.

Среди команд современных ЭВМ практически не встречаются четырехадресные. Мало и трехадресных команд (см. рис. 1.5, *г*), так как результат операции почти всегда можно записать на место одного из уже использованных операндов. Наибольшее распространение в микроЭВМ получили одноадресные и безадресные команды (см. рис. 1.8, *б, в*), позволяющие построить простой процессор, например, показанный на рис. 1.11. Но прежде чем перейти к обсуждению таких коротких команд, ответим на ряд вопросов, которые, как нам кажется, могут возникнуть у любознательного читателя.

Не потребуется ли хранить в разных ячейках памяти отдельные поля (или части полей) трех-, двух- или одноадресных команд и как работать с подобными командами?

Чем объясняется появление в процессоре, изображенном на рис. 1.11, дополнительных регистров (регистра адреса, счетчика команд и регистра данных)? Почему эти регистры связаны между собой так, как показано на рис. 1.11, а не иначе?

По нашему мнению, ответы на эти и другие вопросы лучше всего дать в форме обсуждения целесообразности использования различных вариантов процессора, реализующего четырехадресные команды.

На рис. 1.10, *а*, приведена одна из возможных структур обработки четырехадресной команды, размещенной в нескольких ячейках памяти. В ней:

команды и данные расположены в памяти, состоящей из 16-битовых ячеек;

на четырехадресную команду отводятся пять ячеек памяти, в первой из которых находится код операции команды, во второй — адрес первого операнда, в третьей — адрес второго операнда, в четвертой — адрес результата и в пятой — адрес следующей команды;

каждый операнд (результат) занимает одну ячейку памяти;

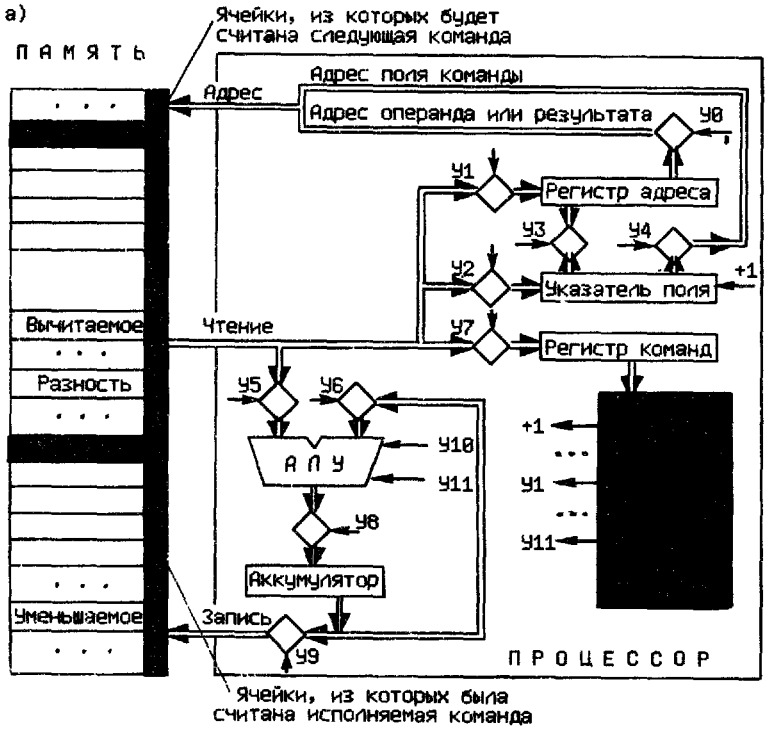
в процессоре используются 16-разрядное АЛУ и 16-разрядные регистры.

Перед обсуждением достоинств и (или) недостатков нового расположения полей команды в памяти ЭВМ и назначения новых регистров, вернемся к структуре, изображенной на рис. 1.9. В ней при опросе последнего адресного поля исполняемой команды (сигнал *У4*) в память пересылался адрес той ячейки, где размещались сразу все поля новой команды. Эта новая команда целиком переписывалась в регистр команд (сигнал *У7*), и процессор получал всю необходимую информацию для ее выполнения. Распределив же поля команды по разным ячейкам памяти (рис. 1.10, *а*), мы усложняем алгоритм исполнения команды. Теперь в последнем адресном поле будет храниться адрес не всей команды, а лишь ее первого поля. А где же взять адреса следующих полей?

Так как адрес каждого следующего поля отличается от адреса предыдущего поля на единицу, то для последовательного получения и хранения этих адресов можно ввести в состав процессора специальный регистр, содержимое которого будет наращиваться на единицу после выборки из памяти очередного поля команды. Перед началом выполнения любой команды в этом регистре (назовем его [illegible]) должен храниться адрес ее первого поля, записанный туда при завершении предыдущей команды. После выборки из памяти содержимого первого поля (кода операции команды) и пересылки его в регистр команд содержимое указателя поля наращивается на единицу (сигнал +1), т. е. становится равным адресу второго поля исполняемой команды. Теперь можно пересылать в память этот адрес для считывания содержимого второго поля, т. е. адреса первого операнда.

Но получив на шине «Чтение» адрес первого операнда, мы не можем непосредственно использовать его для чтения самого операнда: как бы мы не связывали между собой выходы шины «Чтение» и входы адресной шины, нам не удастся одновременно получить на выходе памяти и адрес операнда, и его содержимое. Поэтому адрес операнда надо сохранить в каком-либо регистре процессора (назовем его регистром адреса), отключить адресную шину от указателя поля, а уж затем (например, через время *T*) подключить ее к регистру адреса в целях получения значения первого операнда.

Одновременно с получением значения первого операнда можно подать на указатель поля сигнал +1 и создать в нем адрес второго операнда. Когда же завершится обработка первого операнда, устройство управления подключит указатель поля к шине адреса, регистр адреса к шине «Чтение» и обеспечит тем самым пересылку в этот регистр адреса второго операнда. Получив и обработав второй операнд, а также увеличив на единицу содержимое указателя



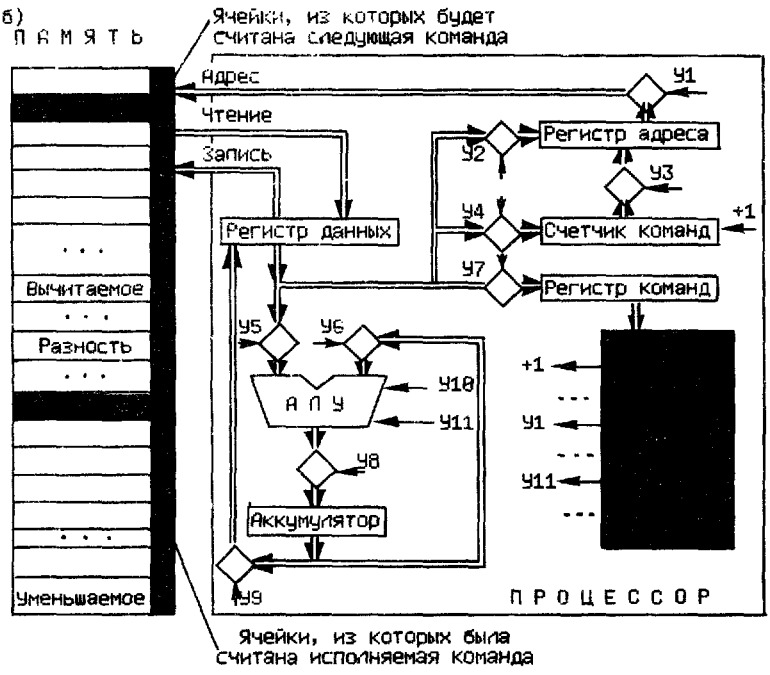


Рис. 1.10. Процессор, работающий с четырехадресной командой, размещенной в пяти соседних ячейках памяти:

а — вариант 1; *б* — вариант 2

поля, процессор переходит к действиям, связанным с пересылкой в память результата операции. При этом содержимое указателя поля вновь увеличивается на единицу, т. е. он будет указывать на ячейку памяти, где содержится адрес первого поля следующей команды.

Когда из указателя поля считывается и пересылается на адресную шину адрес последнего поля исполняемой команды, то содержимое этого поля должно быть переписано из памяти в тот же указатель поля. Поскольку эти действия нельзя выполнить одновременно, содержимое последнего поля следует временно записать в регистр адреса (куда записывались и предыдущие адресные поля), а затем (например, через время *T*) переписать его в указатель поля (сигнал *У3*).

Вернемся теперь к операции, закодированной кодом операции исполняемой команды, и начнем ее рассмотрение с момента, когда в аккумуляторе уже хранится значение первого операнда (например, уменьшаемого), а в регистре адреса — адрес ячейки памяти со значением второго операнда (вычитаемого). Подачей сигналов *У0, У5, У6* и *У11* обеспечиваются:

пересылку в память адреса вычитаемого (сигнал *У0*) и приказа на его чтение;

подключение входов АЛУ к аккумулятору (сигнал *У6*) и шине «Чтение» (сигнал *У5*), на которой должно появиться значение вычитаемого;

настройку АЛУ на выполнение операции вычитания (сигнал *У11*).

Выше отмечалось, что время доступа к ячейке памяти не зависит от адреса этой ячейки, но из-за сложности организации памяти это время больше времени доступа к регистрам процессора. Поэтому уменьшаемое появляется на входе АЛУ раньше вычитаемого. Для того чтобы не увеличивать длительность тактовых импульсов и не снижать тем самым быстродействия ЭВМ, а также для обеспечения возможности использования блоков памяти с различным временем доступа к информации целесообразно включить в состав процессора буфер. В этот буфер (назовем его регистром данных) будет пересылаться содержимое адресуемой ячейки памяти перед его использованием в процессоре (рис. 1.10, *б*).

В новой структуре процессора операция вычитания разбивается на два этапа.

1. Пересылка в память адреса вычитаемого (сигнал *У1*), приказа на чтение и запись вычитаемого в регистр данных. При этом устройство управления не переходит к второму этапу до тех пор, пока не получит из памяти сигнала о выдаче на шину «Чтение» содержимого адресуемой ячейки (значения вычитаемого).

2. Настройка АЛУ на выполнение операции вычитания (сигнал *У11*) и подключение входов АЛУ к аккумулятору (сигнал *У6*) и регистру данных (сигнал *У5*).

Следует отметить, что наличие регистра данных позволило произвести несколько целесообразных изменений структуры процессора. Теперь с памятью связаны лишь два регистра: регистр адреса (через который посылаются в память как адреса полей команды, так и адреса операндов или результата) и регистр данных (который является буфером во всех операциях обмена информацией между памятью и остальными регистрами процессора). Кроме того, указатель поля получил не совсем точное, но очень распространенное название счетчика команд.

Обсуждение структур процессоров, изображенных на рис. 1.10, *а, б*, показало, что четырехадресную команду можно размещать в нескольких ячейках памяти, уменьшая тем самым разрядность этих ячеек, регистров процессора, шин и других элементов, а следовательно, габариты и стоимость всей ЭВМ. Однако общее количество битов, отводимых для хранения такой команды в памяти ЭВМ, не изменяется. Их число можно уменьшить лишь за счет сокращения полей команды.

Если команды программы размещать в памяти ЭВМ друг за другом (а не в произвольной последовательности), то адрес следующей команды чаще всего будет отличаться от адреса исполняемой команды (или ее последнего поля) лишь на единицу, а добавление единицы к текущему адресу можно возложить на ЭВМ (счетчик команд на рис. 1.11). Это позволяет сократить длину команды (изъять из ее содержимого адрес следующей команды), но приводит к необходимости использования специальных команд перехода, размещаемых в тех местах программы, где может потребоваться изменение естественной последовательности выполнения команд в зависимости от результата вычислений. Обычно команды перехода — одноадресные команды (см. рис. 1.8, *б*), где код операции оговаривает проверяемое условие (знак результата предыдущей операции, наличие переноса из старшего разряда и т. п.), а адрес — адрес команды, к которой нужно перейти, если условие выполняется (при невыполнении условия выбирается команда, расположенная вслед за командой перехода).

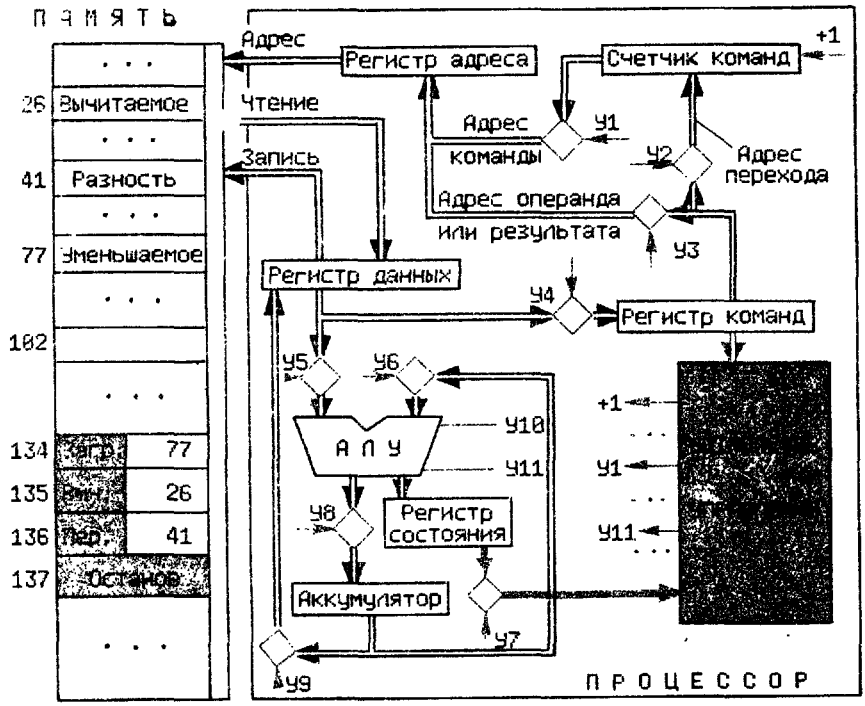


Рис. 1.11. Простой процессор, работающий с одноадресными и безадресными командами

Теперь выясним, всегда ли нужны остальные адресные поля команды.

Анализ различных программ для ЭВМ показывает, что во многих случаях результат выполнения предыдущей команды используется как операнд в следующей. Если результат выполнения команды не пересылать в память, а сохранять, например, в аккумуляторе, то можно обойтись следующими одноадресными командами:

командами арифметической (логической) обработки, которые дают приказ на выполнение какой-либо арифметической или логической операции, используя в качестве операндов содержимое аккумулятора и содержимое адресуемой ячейки памяти (точнее, содержимое регистра данных, куда переписывается содержимое адресуемой ячейки);

командами пересылки, которые дают приказ на обмен информацией между аккумулятором и памятью (через регистр данных), т. е. на загрузку аккумулятора содержимым адресуемой ячейки памяти или запись в эту ячейку содержимого аккумулятора (результата вычислений);

командами передачи управления, обеспечивающими условный или безусловный переход к адресуемой ячейке памяти (т. е. к команде, которая должна быть помещена в процессор при выполнении какого-либо условия или вне зависимости от результата предыдущих вычислений), обращение к подпрограмме и выход из нее;

командами ввода-вывода.

Как же с помощью таких команд будет выполняться, например, суммирование по формуле вида $A=B+C+E+F+G+H$? Для этого надо сначала загрузить в аккумулятор значение слагаемого B, затем выполнить пять оди-

наковых команд сложения и, наконец, команду пересылки содержимого аккумулятора в ячейку, выделенную для суммы А:

Команды программы		Содержимое аккумулятора в процессе выполнения программы
Загрузить	Адрес слагаемого B	B
Сложить	Адрес слагаемого C	B+C
Сложить	Адрес слагаемого E	B+C+E
Сложить	Адрес слагаемого F	B+C+E+F
Сложить	Адрес слагаемого G	B+C+E+F+G
Сложить	Адрес слагаемого H	B+C+E+F+G+H
Переслать	Адрес результата A	B+C+E+F+G+H
Останов		B+C+E+F+G+H

(Аналогичным образом Вы работаете с микрокалькулятором, когда, введя очередное слагаемое и нажав клавишу «+», добавляете это слагаемое к накопленной ранее сумме).

Отметим, что последняя из команд рассмотренной программы вообще не имеет адреса. Эта команда адресуется к устройству управления, а не к имеющей множество ячеек памяти. Подобные команды, предназначенные для работы с конкретными устройствами (аккумулятором, устройством управления и т. п.), называются безадресными командами или командами с неявной адресацией. Они позволяют сформировать приказы на обнуление аккумулятора, добавление к его содержимому константы (+1 или —1), перевод ЭВМ в состояние разрешения прерывания выполняемой программы и т. п.

Процессоры, реализующие программы из одноадресных и безадресных команд (см. рис. 1.8, *б, в*), являются простейшими и позволяют создавать наиболее дешевые ЭВМ. В одной из возможных структур такого процессора (см. рис. 1.11) используются уже знакомые нам устройства:

арифметико-логическое устройство производит операции над двумя 16-разрядными величинами в целях получения 16-разрядного результата и выработки ряда признаков (результат меньше нуля, равен нулю или больше нуля, был перенос из старшего разряда результата);

регистр состояния — 4-битовый регистр, в котором хранятся признаки результата последней операции, используемые командами перехода;

аккумулятор — 16-разрядный регистр, в котором размещаются подлежащие обработке данные или результат обработки;

регистр команд — 16-битовый регистр, служащий для размещения исполняемой команды;

регистр адреса — 12-битовый регистр, содержащий адрес ячейки памяти, из которой будет считана команда (операнд) или записан результат обработки;

регистр данных — 16-битовый регистр, используемый в качестве буфера между памятью и остальными регистрами процессора; через него пересылаются в процессор команды (операнды) и передаются в память результаты обработки;

счетчик команд — 12-битовый регистр, содержимое которого увеличивается на единицу в момент выборки из памяти исполняемой команды и, если выбрана команда перехода, может быть заменено на содержимое адресной

части команды перехода; в конце цикла исполнения команды в счетчике команд всегда хранится адрес той команды, которая должна исполняться вслед за текущей, а это может быть как следующая по порядку команда, так и команда, к которой требуется перейти при выполнении условий, заданных кодом операции команды перехода.

Для решения задачи на такой ЭВМ необходимо:

1) через устройства ввода информации загрузить в память ЭВМ программу решения задачи (алгоритм, написанный на языке ЭВМ) и исходные данные (это делается различными способами, о которых будет рассказано в следующих главах); и программа, и данные могут быть размещены в любой области памяти, начиная с ячейки 0 или другой ячейки с любым адресом (сначала программа, а затем данные, или наоборот).

2) «сообщить» процессору адрес ячейки памяти, в которой размещена первая команда программы, для чего занести адрес этой ячейки в счетчик команд (об одном из способов загрузки пускового адреса будет рассказано в параграфе 3.1);

3) нажать кнопку ПУСК, что приведет к передаче в память адреса первой команды программы и к пересылке ее содержимого из памяти в регистр команд; с этого момента процессор начнет выполнять последовательность достаточно простых операций, показанных на схеме алгоритма выполнения команды (рис. 1.12).

Как видно из схемы алгоритма, два первых действия (блоки *1* и *2*) выполняются для каждой команды. Эти действия (а также действия по определению типа команды — блоки *3, 6* и *9*) обычно называют «Выборкой команды». Последующие действия алгоритма («Исполнение команды») полностью зависят от того, какая это команда.

Рассмотрим исполнение команды, расположенной в ячейке 135 на рис. 1.11. Оно начинается в момент, когда в аккумуляторе хранится значение уменьшаемого, записанное туда при выполнении предыдущей команды, а в счетчике команд — адрес 135.

1. Адрес, содержащийся в счетчике команд, переписывается в регистр адреса и далее в память ЭВМ (сигнал *У1*), а оттуда считывается команда *«Выч. 26»*, помещаемая в регистр команд (сигнал *У4*).

2. Счетчик команд наращивается на единицу (сигнал +1), чтобы он указывал на команду (адрес 136), расположенную в памяти вслед за выбранной командой.

3. Устройство управления считывает из регистра команд код исполняемой команды, выясняет, что это команда вычитания, и переходит к ее выполнению.

4. Из регистра команд считывается (сигнал *У3*) и пересылается в память (через регистр адреса) адрес вычитаемого (26), а из памяти переписывается в регистр данных численное значение вычитаемого.

5. Выполняется операция вычитания содержимого регистра данных из содержимого аккумулятора (сигналы *У5, У6* и *У11*), а затем полученная разность переписывается с выхода АЛУ в аккумулятор (сигнал *У8*), заменяя значение уменьшаемого.

Легко заметить, что процесс выполнения команды сводится к определению последовательности открывания и закрывания вентильных схем. Описание того, какую вентильную схему и когда открывать, составляет программу для машины, система команд которой включает команду «Открыть вентильную схему». Такой машиной и является устройство управления ЭВМ, в котором хранятся программы реализации всех команд ЭВМ. Эти программы обычно называются микропрограммами, а их команды — микрокомандами. Простейшая микрокоманда состоит из набора битов, каждый из которых управляет одной из вентильных схем процессора: единица означает, что вен-

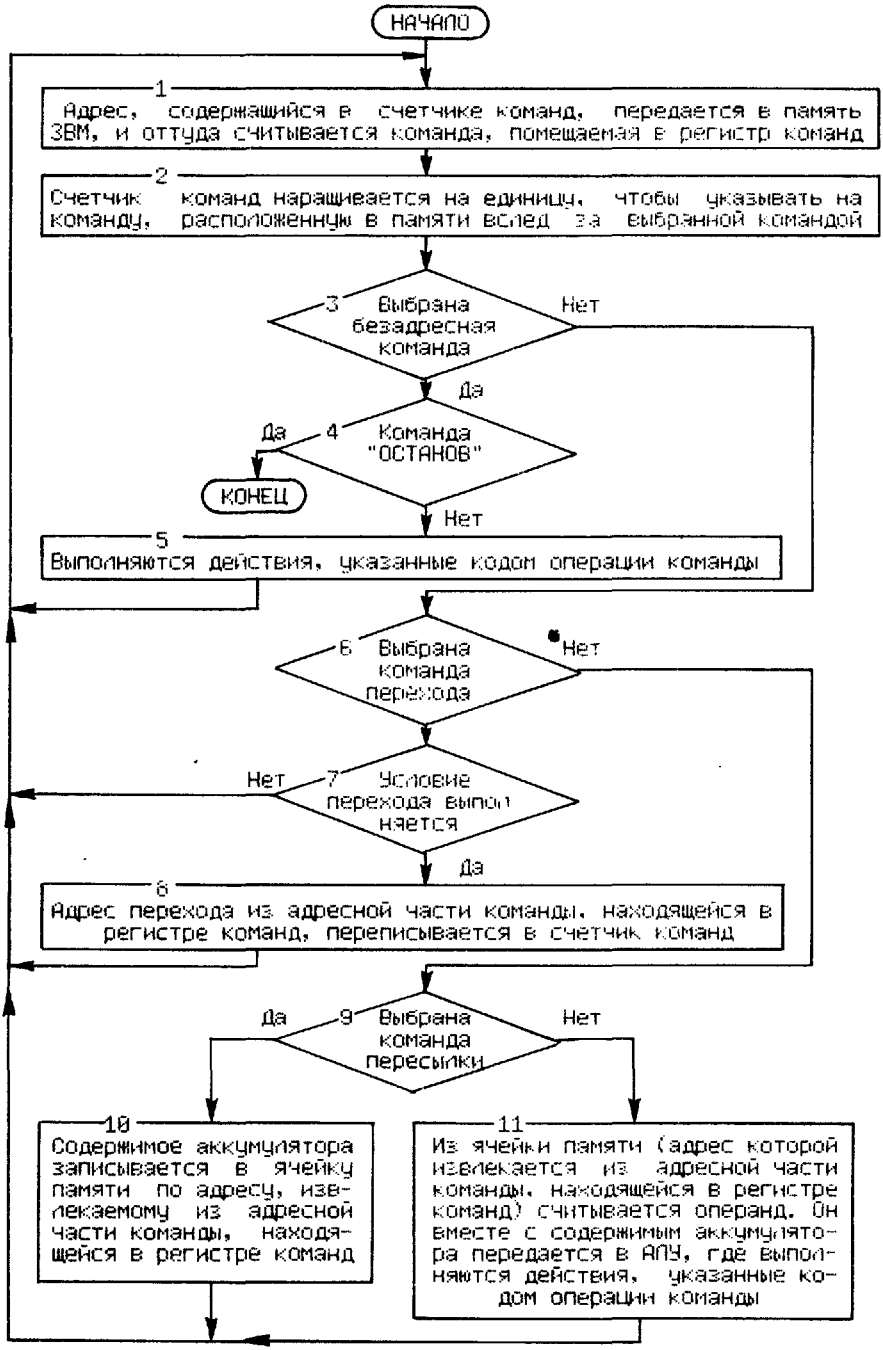


Рис. 1.12. Схема алгоритма выполнения команд простого процессора

тильная схема открыта, нуль — закрыта. С микропрограммным уровнем можно ознакомиться в гл. 4.

В заключение отметим, что процессор, изображенный на рис. 1.11, не является единственно возможным процессором, используемым для реализации одноадресных команд. Так, сумму произведений

$$A \cdot B + C \cdot E + F \cdot H + G \cdot K + \ldots$$

лучше вычислять с помощью процессора, имеющего два аккумулятора (один — для вычисления произведений, а другой — для накопления суммы). Это позволит исключить пересылку в память промежуточных результатов, т. е. ускорить вычисления и уменьшить число используемых ячеек памяти. Можно привести примеры, где целесообразно использовать процессор с большим числом аккумуляторов или других регистров (например, индексных). Процессоры, имеющие более двух аккумуляторов, называются процессорами с регистрами общего назначения (с РОНами).

Резюме

1. Структурно любая ЭВМ состоит из процессора, памяти и устройств ввода-вывода.

2. Память ЭВМ состоит из ячеек одинакового размера, каждая из которых имеет свой уникальный номер — адрес и может хранить команду или число. Количество элементов памяти (число битов), из которых составлена каждая ячейка, определяет разрядность машинных слов — команд и данных или их частей.

3. Все команды в качестве одной (может быть единственной) части содержат код операции, задающий действие, исполняемое по этой команде. Кроме кода операции команда может иметь несколько адресных частей, содержащих адреса операндов, результата и следующей команды.

4. В микроЭВМ используются безадресные, одноадресные и реже двухадресные команды. В одноадресных командах один из операндов выбирается из специального регистра — аккумулятора. В него же заносится и результат операции. Безадресные команды или задают какое-либо действие с устройствами ЭВМ (например, останов), или используются для работы с операндами, имеющими фиксированное расположение (чаще всего с аккумулятором).

5. Процессор — это совокупность устройства управления, арифметико-логического устройства и связанных с ними регистров. Он организует процесс обработки информации путем выборки и последовательного выполнения команд программы, находящейся в памяти ЭВМ.

6. Арифметико-логическое устройство выполняет ряд арифметических и логических операций над одним или двумя операндами, пересылаемыми в него из памяти и (или) регистров процессора. Результат операции пересылается в один из регистров процессора.

7. Устройство управления, используя регистры процессора, управляет обменом информации между памятью, устройствами ввода-вывода и АЛУ, а также выполнением операций в АЛУ.

8. Регистры процессора служат для хранения промежуточных результатов вычислений и различной управляющей информации. Наиболее важные из них — это регистр команд (содержит исполняемую команду) и счетчик команд (содержит адрес следующей команды).

9. В процессе работы ЭВМ последовательно выполняет набор достаточно простых операций: выборку команды, определение ее типа, исполнение команды и определение адреса следующей команды. Разнообразие же решаемых на ЭВМ задач определяется не ее устройством, а программой и возможностями подключаемых к ЭВМ устройств ввода-вывода.

Глава 2

БАЗОВАЯ ЭВМ

2.1. Назначение и структура базовой ЭВМ

Вычислительные машины существенно различаются по возможностям, размерам и стоимости. Некоторые машины могут выполнять очень ограниченное число простых операций, в то время как другие — сотни различных операций (команд). Теоретически для решения конкретной задачи можно использовать любую ЭВМ. Малая будет решать задачу путем выполнения большого числа очень простых команд, в связи с чем может потребоваться большой интервал времени для получения решения. Большая ЭВМ, способная выполнять множество операций, решит ту же задачу значительно быстрее.

В этой главе рассмотрена простая гипотетическая машина, обладающая типичными чертами многих конкретных микроЭВМ. Знание принципов построения и функционирования этой ЭВМ является хорошей базой для освоения микропроцессорных систем любых типов и моделей, поэтому она названа базовой ЭВМ. Естественно, начинать изучение ЭВМ целесообразно с подобной машины низкого уровня, что можно делать практически, используя ее модели, построенные для разных типов персональных ЭВМ. При построении базовой ЭВМ за прототип выбраны ЭВМ «Электроника 100» и «Саратов», а также схожие с ними ЭВМ типа PDP-8, однокристальный микропроцессор IM 6100 и персональная ЭВМ DECmate 11 [21, 37, 40].

На рис. 2.1 приведена упрощенная структура базовой ЭВМ. Это одноадресная машина, работающая с 16-разрядными словами. В ней реализованы два вида адресации операндов: прямая и косвенная. Рассмотрим составные части базовой ЭВМ, не касаясь пока устройств ввода-вывода (УВВ) и пульта управления (ПУ).

Память. Состоит память из 2048 ячеек по 16 бит с адресами 0, 1, ..., 2046, 2047. Одну ячейку (или слово) можно использовать для хранения одного двоичного числа или одной команды программы, закодированной в двоичном коде. Пользователь должен сам определить, какая область памяти будет использоваться для данных и какая — для программы. При этом надо учитывать, что восемь ячеек памяти с адресами 008, ..., 00F несколько отличаются от остальных. Эти ячейки называются индексными и их лучше использовать в циклических программах.

Команды исходной программы обычно размещаются пользователем в соседних ячейках памяти. Машина считывает одну команду из памяти и выполняет указанную в ней операцию. Затем она считывает следующую команду из последующей ячейки и выполняет следующую операцию и т. д.

Процессор. В состав процессора входят:

шесть регистров (16-разрядные регистр команд, аккумулятор и регистр данных, 11-разрядные счетчик команд и регистр адреса, 1-разрядный регистр переноса);

арифметико-логическое устройство, выполняющее операции инвертирования любого операнда (или двух операндов), сложения двух 16-разрядных операндов (или их инвертированых значений), логического умножения двух 16-разрядных операндов, добавления к результату единицы и участвующее в организации циклических сдвигов содержимого аккумулятора и регистра переноса;

микропрограммное устройство управления, хранящее в своей памяти микропрограммы выполнения 28 команд (см. табл. 2.4), четырех пультовых

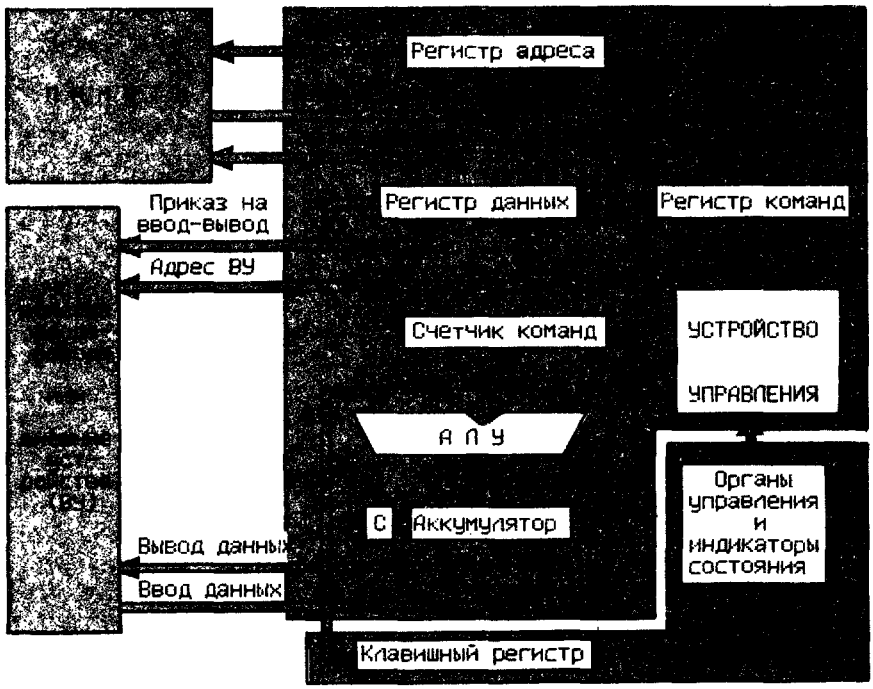


Рис. 2.1. Структура базовой ЭВМ

операций, реакции на сигналы прерывания выполняемой программы от устройств ввода-вывода и ряд других микропрограмм.

Рассмотрим подробнее назначение всех регистров процессора и АЛУ базовой ЭВМ.

Счетчик команд (СК). Он служит для организации обращения к ячейкам памяти, в которых хранится программа. В конце каждого цикла исполнения команды счетчик указывает адрес ячейки памяти, содержащей следующую команду программы. Так как команды чаще всего заносятся в последовательные ячейки, а при выполнении текущей команды содержимое счетчика увеличивается на единицу, то он автоматически указывает адрес следующей команды. В некоторых случаях содержимое счетчика команд может быть изменено самой программой (например, при выполнении команд переходов). Таким образом осуществляется передача управления другой части программы. В базовой ЭВМ используется 11-разрядный счетчик команд, который может формировать адреса любого из 2048 слов памяти.

Регистр команд (РК). Этот 16-разрядный регистр используется для хранения команды, непосредственно выполняемой машиной. Код операции команды (см., например, рис. 1.8, 1.11 или 2.3) пересылается из РК в устройство управления ЭВМ и декодируется. После этого происходят действия по реализации команды: считывание операнда и (или) выполнение операции, предписываемой командой.

Регистр адреса (РА). Он содержит значение исполнительного адреса ячейки памяти и состоит из 11 разрядов для адресации к 2048 ячейкам памяти. Если ЭВМ осуществляет выборку команды, то в РА пересылается содержимое счетчика команды СК для указания адреса ячейки, где хранится команда. Если ЭВМ производит выборку данных, то адрес может поступить из регистра команд.

Регистр данных (РД). Используется он для временного хранения 16-разрядных слов при обмене информацией между памятью и процессором. При считывании команды, числа или символа из памяти в процессор это слово сначала попадает в регистр данных, а потом пересылается либо в регистр команд (команды), либо в другие регистры процессора. При пересылке данных из процессора в память они сначала помещаются в РД и лишь затем записываются в нужную ячейку памяти.

Когда информационное слово находится в регистре данных, оно доступно для осуществления арифметических или логических операций. Содержимое РД может быть, например, сложено с содержимым аккумулятора, а полученный результат занесен в аккумулятор.

Аккумулятор (А). Регистр А является одним из главных элементов процессора. Машина может выполнять арифметическую или логическую операцию только над двумя операндами одновременно. Обычно первый операнд извлекается из памяти в регистр данных, в то время как второй находится в аккумуляторе. Операция, задаваемая командой, выполняется над содержимыми РД и А, и результат операции помещается в аккумулятор.

Машина может проверить результат в аккумуляторе. В зависимости от результата проверки она может принимать различные решения. Здесь используется 16-разрядный аккумулятор. Всякий раз, когда при операции двоичного сложения возникает переполнение в старшем разряде, перенос теряется. Однако его можно записать в одноразрядный регистр переноса.

Регистр переноса (С). Это одноразрядный регистр, выступающий в качестве продолжения аккумулятора и заполняющийся при его переполнении. Этот регистр также используется для организации циклических сдвигов. Состояние регистра переноса может проверяться для принятия решений.

Арифметико-логическое устройство. АЛУ базовой ЭВМ позволяет выполнять такие арифметические операции, как сложение и вычитание с учетом переноса (содержимого регистра С), полученного в результате осуществления предыдущей операции. Кроме того, оно способно выполнять операции логического умножения (операцию И), инвертирования, циклического сдвига и наращивания аккумулятора на 1.

Если с помощью рассматриваемой ЭВМ надо, например, получить сумму чисел 53 и 106, то это можно сделать по программе, приведенной в табл. 2.1. Хотя и команды, и данные должны быть закодированы в двоичной форме (единственной форме, которую понимает ЭВМ), эта программа для простоты написана с символическим обозначением команд (подробнее о символическом кодировании см. в параграфе 2.2).

Используются следующие команды:

CLA (**Cl**ear **A**ccumulators — очистить аккумулятор) — по этой команде производится установка аккумулятора в нуль;

Таблица 2.1

Программа вычисления суммы двух чисел

Ячейки памяти		Комментарии
Адрес	Содержимое	
20	0053	Первое слагаемое
21	0106	Второе слагаемое
22	0000	Ячейка, предназначенная для результата
23	CLA	Аккумулятор содержит 0000 (очистка)
24	ADD 20	Аккумулятор содержит 53
25	ADD 21	Аккумулятор содержит 159 (53+106)
26	MOV 22	В ячейку с адресом 22 записывается 159
27	HLT	Останов машины (прекращение выборки команд)

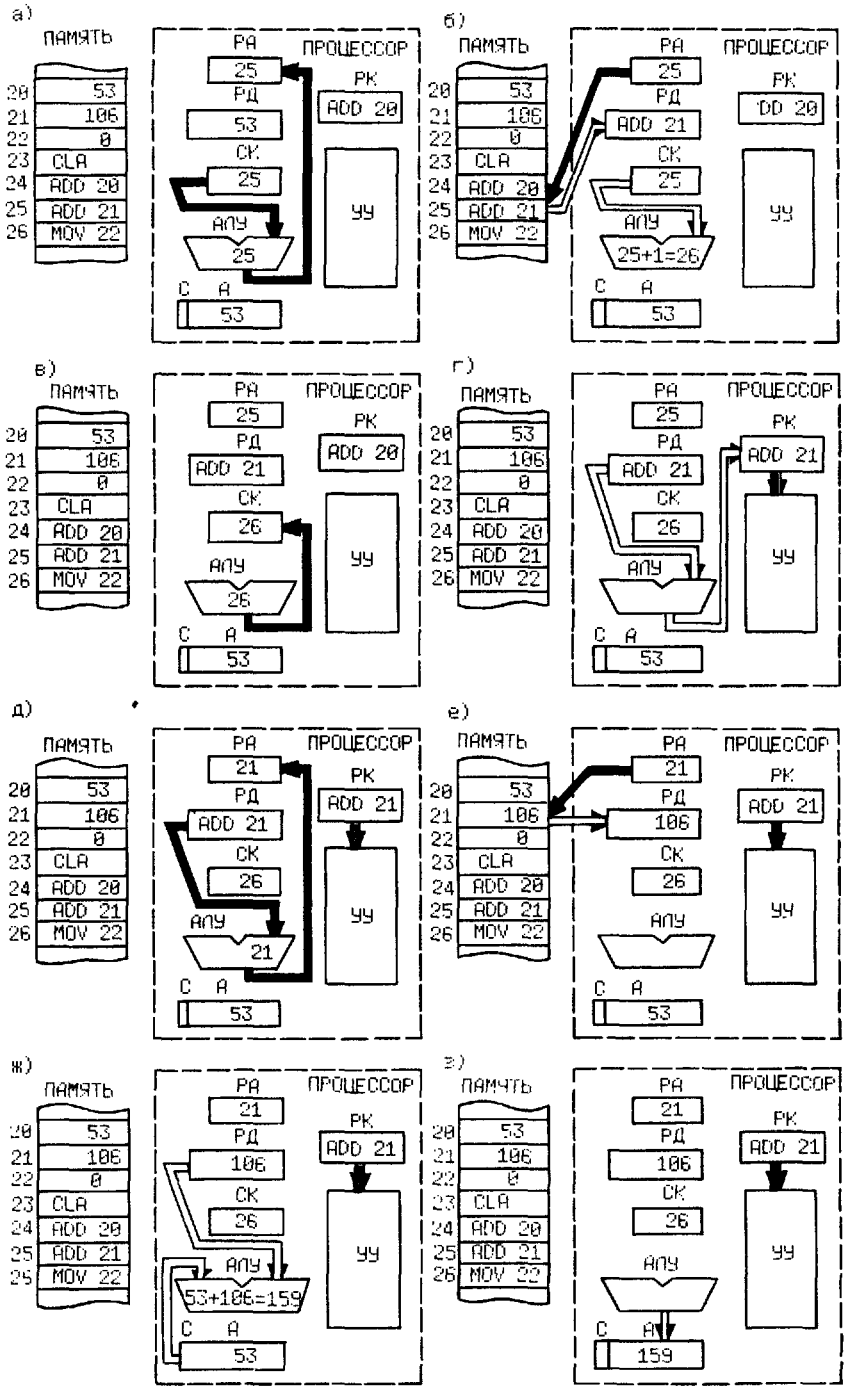


Рис. 2.2. Выполнение команды ADD 21

ADD (**ADD** — сложить) — по этой команде содержимое ячейки с номером, написанным вслед за ADD, складывается с содержимым аккумулятора и результат остается в аккумуляторе;

MOV (**MOV**e — переслать) — по этой команде содержимое аккумулятора пересылается в ячейку с номером, написанным вслед за MOV; содержимое аккумулятора при этом сохраняется;

HLT (**H**a**LT** — стоп).

Программа записана в память ЭВМ начиная с ячейки 23, а числовые данные - с ячейки 20. Ячейка с номером 22 отведена для записи суммы. Для выполнения программы необходимо установить в счетчик команд число 23 и пустить ЭВМ. Тогда выполнение программы начнется с чтения содержимого ячейки 23. Команда CLA установит аккумулятор в нуль. При выполнении этой команды содержимое счетчика команд увеличится на 1 и следующая команда будет считываться из ячейки 24. Это команда ADD 20 складывает содержимое ячейки 20 с содержимым аккумулятора, т. е. 53 с 0. По окончании выполнения команды в аккумуляторе содержится число 53, а в счетчике команд — 25 (он опять наращивается на 1).

Следующая команда читается из ячейки 25, выполняется суммирование содержимых ячейки 21 и аккумулятора, т. е. 106 и 53. По окончании команды в аккумуляторе содержится сумма $106+53=159$, а в счетчике команд — 26. Следующая команда — MOV 22 пересылает содержимое аккумулятора в ячейку с адресом 22. Теперь в этой ячейке и аккумуляторе содержится число 159, а в счетчике команд — 27. Из ячейки 27 выбирается команда HLT, счетчик команд наращивается на 1 (в нем содержится число 28), но по команде HLT прекращается выборка команд, т. е фиксируется окончание выполнения программы (содержимое ячейки 28 не выбирается в регистр команд и не интерпретируется как команда).

Рассмотрим теперь подробнее выполнение одной из команд этой программы, например команды ADD 21. Перед ее выполнением в регистре команд хранится предыдущая команда (ADD 20), в регистре адреса — адрес ее операнда (20), в регистре данных и аккумуляторе — значение этого операнда (53) и в счетчике команд — адрес рассматриваемой команды (25).

Сначала адрес команды пересылается из счетчика команд в регистр адреса (рис. 2.2, *а*). Здесь следует отметить, что в базовой ЭВМ все пересылки между регистрами выполняются через АЛУ (при необходимости пересылаемое значение инвертируется, к нему добавляется единица и т. п.). Затем из памяти в регистр данных выбирается содержимое ячейки памяти, адрес которой расположен в регистре адреса, и производится увеличение на единицу содержимого счетчика команд (рис. 2.2, *б, в*). Наконец, содержимое регистра данных пересылается в регистр команд, и устройство управления начинает анализировать это содержимое, т. е. команду ADD 21 (рис. 2.2, *г*). Перечисленные действия относятся к «Выборке команды».

Расшифровав код операции команды ADD 21, устройство управления пересылает ее адресную часть (адрес 21) в регистр адреса (рис. 2.2, *д*). Пересылка осуществляется из регистра данных, где еще сохраняется копия команды. Затем из памяти в регистр данных считывается содержимое ячейки памяти, адрес которой расположен в регистре адреса (рис. 2.2, *е*). Это действие аналогично действию на рис. 2.2, *б*, но устройство управления будет теперь рассматривать содержимое регистра данных не как команду, а как слагаемое, которое на следующем шаге складывается с содержимым аккумулятора (рис. 2.2, *ж*). Наконец, полученная в АЛУ сумма пересылается в аккумулятор (рис. 2.2, *з*) и тем самым завершается «Исполнение команды». Набор и (или) порядок выполнения операций в этом цикле уникальны для каждой из команд ЭВМ и будут рассмотрены ниже.

2.2. Кодирование программ и система команд

Все операции в ЭВМ выполняются над электрическими сигналами, закодированными двоичной цифрой 0 или 1. Поэтому перед занесением в память данные и команды должны быть вручную или аппаратно преобразованы в двоичную форму. Однако при написании программы команды могут быть записаны в любой другой форме (например, мнемонической, как в табл. 2.1). В общем случае программа может кодироваться четырьмя способами: в двоичной, восьмеричной, шестнадцатеричной и символической или мнемонической форме.

Двоичная форма команды. Эта форма является единственной, которую понимает машина. Каждой ЭВМ присуща система команд в двоичном коде, которая понятна ей. Регистр команд, куда помещается команда из памяти, связан электрическими цепями с устройством управления ЭВМ, где производится декодирование команды и инициируются сигналы по реализации команды. В табл. 2.2 приведено содержимое части памяти ЭВМ, где хранятся программа и данные для суммирования двух чисел из примера параграфа 2.1. Левый столбец – это адреса ячеек в двоичной форме. Правый столбец представляет содержимое ячеек в двоичной форме. Адрес имеет 11 разрядов (память ЭВМ состоит из $2048 = 2^{11}$ ячеек), а каждая из ячеек – по 16 разрядов. Для удобства чтения двоичных чисел и их перевода в шестнадцатеричную систему они разбиты на тетрады.

Восьмеричное и шестнадцатеричное кодирование. Из предыдущего примера ясно, что записывать программу в двоичном коде очень неудобно и утомительно, если учесть, что средняя по сложности программа состоит из нескольких сотен или тысяч нулей и единиц. Для сокращения записи целесообразно использовать систему счисления с основанием $2^3 = 8$ или $2^4 = 16$ (см. параграф 1.3). Выбор для кодирования команд той или иной системы счисления в основном связан с форматом команд ЭВМ. В нашем случае удобнее шестнадцатеричная система, так как большинство полей в командах базовой ЭВМ (см. рис. 2.2) равно или кратно 4. В табл. 2.3 приведена та же программа, что и в табл. 2.1 и 2.2, только закодированная в шестнадцатеричной системе. Для программиста такое кодирование более удобно, чем двоичное.

Мнемоническое (символическое) кодирование. Шестнадцатеричное (или восьмеричное) кодирование имеет очевидное преимущество перед двоичным. Однако для длинных программ и оно неудобно. Программист должен выучить наизусть шестнадцатеричные коды всех команд, использующихся в машине (в некоторых ЭВМ их более сотни). Чтобы упростить процесс написания, отладки и чтения программы, предложен мнемонический или символический код: каждая команда представляется простым двух-, трех- или четырехбуквенным мнемоническим символом. Мнемонические символы значительно легче связать с машинными операциями, так как их можно выбирать таким образом, чтобы они напоминали название команды. Большинство мнемонических кодов – это сокращения английских названий команд: SUB от **SUB**tract (вычесть), BR от **BR**anch (перейти), BPL от **B**ranch if **PL**us (перейти по положительному числу) и т. п. Намного легче запомнить, что инвертирование аккумулятора (**C**o**M**plement **A**ccumulator) кодируется CMA, чем запомнить двоичный код 1111010000000000 или даже его шестнадцатеричный эквивалент F400. Пример символического кодирования программы был приведен в параграфе 2.1 (см. табл. 2.1).

Хотя символическое кодирование очень удобно для программиста, оно не может быть понято машиной. Единственным языком, понятным машине, является язык двоичных кодов. Следовательно, необходимо транслировать символическую программу в ее двоичный эквивалент. Это можно сделать

Таблица 2.2

Пример двоичного кодирования содержимого памяти

Ячейка	Содержимое
000 0010 0000	0000 0000 0101 0011
000 0010 0001	0000 0001 0000 0110
000 0010 0010	0000 0000 0000 0000
000 0010 0011	1111 0010 0000 0000
000 0010 0100	0100 0000 0010 0000
000 0010 0101	0100 0000 0010 0001
000 0010 0110	0011 0000 0010 0010
000 0010 0111	1111 0000 0000 0000

Таблица 2.3

Пример шестнадцатеричного кодирования содержимого памяти

Ячейка	Содержимое
020	0053
021	0106
022	0000
023	F200
024	4020
025	4021
026	3022
027	F000

вручную, используя таблицы соответствия (вида табл. 2.4). На практике трансляция осуществляется специальной машинной программой.

Классификация команд. ЭВМ способна понимать и выполнять определенный набор команд. При составлении программы программист ограничен этими командами. Количество и тип команд изменяются в зависимости от возможностей и назначения ЭВМ.

В зависимости от того, к каким блокам машины обращается команда или на какие блоки она ссылается, команды можно разделить на три группы: обращения к памяти (адресные команды); обращения к регистрам (регистровые или безадресные); ввода-вывода.

Команды обращения к памяти предписывают машине производить действия с содержимым ячейки памяти, адрес которой указан в адресной части команды. Например, команда ADD 20 из табл. 2.1 является командой обращения к памяти. Она предписывает машине обращение по адресу 20 и использование содержимого этой ячейки в качестве первого операнда. Второй операнд находится в аккумуляторе. Эти два операнда суммируются.

Безадресные команды выполняют различные действия без ссылок на ячейку памяти. Например, команда CLA из табл. 2.1 предписывает машине очистить аккумулятор. Эта команда имеет дело с операндом, расположенным в конкретном месте — в аккумуляторе. Другой пример безадресной команды — команда HLT из табл. 2.1.

Команды ввода-вывода осуществляют обмен данными между ЭВМ и внешними устройствами. В них задаются адрес (название) устройства ввода-вывода и код той операции, которую должно выполнить это устройство (приказ на ввод-вывод).

В табл. 2.4 дан перечень команд базовой ЭВМ. Подробно действия, выполняемые машиной по этим командам, рассмотрены в следующих параграфах настоящей главы. Поясним одно из описаний: (CK) + 1 ==> ==> CK в командах ISZ и TSF. После выполнения этой операции счетчик команд будет указывать не на следующую команду программы, а на команду, расположенную за ней. Это произойдет потому, что после выборки команды ISZ или TSF содержимое счетчика команд уже было увеличено автоматически на единицу.

Существует и другой способ разделения команд на группы. Он основан на учете функций, выполняемых командой. Можно выделить восемь типов команд: пересылок или обмена, арифметические, логические, сдвигов, переходов, обращения к подпрограмме, управления и ввода-вывода.

Форматы команд и способы адресации. В параграфе 1.4 рассматривались различные форматы (структуры) команд. Разработчики базовой ЭВМ выбрали три формата 16-битовых (однословных) команд с 4-битовым кодом опера-

Таблица 2.4

Система команд базовой ЭВМ

НАИМЕНОВАНИЕ	МНЕМОНИКА	КОД	ОПИСАНИЕ
АДРЕСНЫЕ КОМАНДЫ			
ЛОГИЧЕСКОЕ УМНОЖЕНИЕ	AND M	1XXX	(M) & (A) ==> A
ПЕРЕСЫЛКА	MOV M	3XXX	(A) ==> M
СЛОЖЕНИЕ	ADD M	4XXX	(M) + (A) ==> A
СЛОЖЕНИЕ С ПЕРЕНОСОМ	ADC M	5XXX	(M) + (A) + (C) ==> A
ВЫЧИТАНИЕ	SUB M	6XXX	(A) - (M) ==> A
ПЕРЕХОД, ЕСЛИ ПЕРЕНОС	BCS M	8XXX	ЕСЛИ (C)=1, ТО M==> CK
ПЕРЕХОД, ЕСЛИ ПЛЮС	BPL M	9XXX	ЕСЛИ (A)>=0, ТО M==> CK
ПЕРЕХОД, ЕСЛИ МИНУС	BMI M	AXXX	ЕСЛИ (A)<0, ТО M==> CK
ПЕРЕХОД, ЕСЛИ НОЛЬ	BEQ M	BXXX	ЕСЛИ (A)=0, ТО M==> CK
БЕЗУСЛОВНЫЙ ПЕРЕХОД	BR M	CXXX	M==> CK
ПРИРАЩЕНИЕ И ПРОПУСК	ISZ M	0XXX	(M)+1==>M; ЕСЛИ (M)>=0, ТО (CK)+1==> CK
ОБРАЩЕНИЕ К П/ПРОГРАММЕ	JSR M	2XXX	(CK)==>M; M+1==> CK
БЕЗАДРЕСНЫЕ КОМАНДЫ			
ОЧИСТКА АККУМУЛЯТОРА	CLA	F200	0==> A
ОЧИСТКА РЕГ. ПЕРЕНОСА	CLC	F300	0==> C
ИНВЕРСИЯ АККУМУЛЯТОРА	CMA	F400	($\bar{A}$)==> A
ИНВЕРСИЯ РЕГ.ПЕРЕНОСА	CMC	F500	($\bar{C}$)==> C
ЦИКЛИЧЕСКИЙ СДВИГ ВЛЕВО НА 1 РАЗРЯД	ROL	F600	СОДЕРЖИМОЕ A И C СДВИГАЕТСЯ ВЛЕВО, A(15)==>C И C==>A(0)
ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО НА 1 РАЗРЯД	ROR	F700	СОДЕРЖИМОЕ A И C СДВИГАЕТСЯ ВПРАВО, A(0)==>C И C==> A(15)
ИНКРЕМЕНТ АККУМУЛЯТОРА	INC	F800	(A) + 1 ==> A
ДЕКРЕМЕНТ АККУМУЛЯТОРА	DEC	F900	(A) - 1 ==> A
ОСТАНОВ	HLT	F000	
НЕТ ОПЕРАЦИИ	NOP	F100	
РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ	EI	FA00	
ЗАПРЕЩЕНИЕ ПРЕРЫВАНИЯ	DI	FB00	
КОМАНДЫ ВВОДА-ВЫВОДА			
ОЧИСТКА ФЛАГА	CLF B	E0XX	0==>ФЛАГ УСТРОЙСТВА B
ОПРОС ФЛАГА	TSF B	E1XX	ЕСЛИ ФЛАГ УСТРОЙСТВА B РАВЕН 1, ТО (CK)+1==>CK
ВВОД	IN B	E2XX	(B) ==> A
ВЫВОД	OUT B	E3XX	(A) ==> B

Условные обозначения:

(M), (A), (CK), (C), (B) - СОДЕРЖИМОЕ ЯЧЕЙКИ С АДРЕСОМ M, АККУМУЛЯТОРА, СЧЕТЧИКА КОМАНД, РЕГИСТРА ПЕРЕНОСА И РЕГИСТРА ДАННЫХ УСТРОЙСТВА ВВОДА-ВЫВОДА С АДРЕСОМ B;

XXX - АДРЕС ЯЧЕЙКИ ПАМЯТИ;

XX - АДРЕС УСТРОЙСТВА ВВОДА-ВЫВОДА.

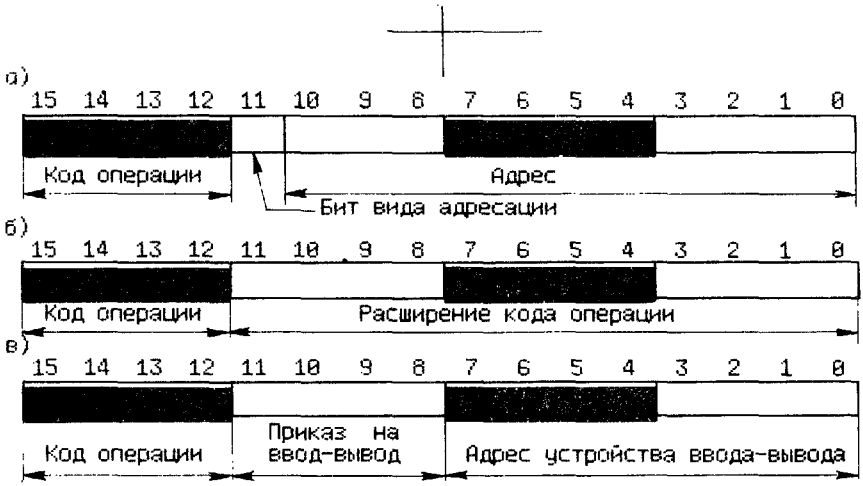


Рис. 2.3. Форматы команд базовой ЭВМ:
а — адресных; *б* — безадресных; *в* — ввода-вывода

ций (рис. 2.3). С помощью 4-битового числа можно закодировать не более чем $2^4=16$ различных операций. Разработчики отвели два кода (1110 и 1111) на команды ввода-вывода и безадресные команды, а так как в этих командах либо используется меньшая по длине адресная часть (8-битовый адрес устройства ввода-вывода на рис. 2.3, *в*), либо эта часть вообще отсутствует (рис. 2.3, *б*), то появилась возможность иметь до $2^4=16$ команд ввода-вывода (4-битовый приказ на ввод-вывод) и до $2^{12}=4096$ безадресных команд (12-битовое расширение кода операции).

В командах обращения к памяти на адрес отведено 11 бит, что позволяет осуществить прямое адресование всех 2048 (2^{11}) ячеек памяти базовой ЭВМ. Однако встречаются приложения, когда в команде целесообразнее разме-

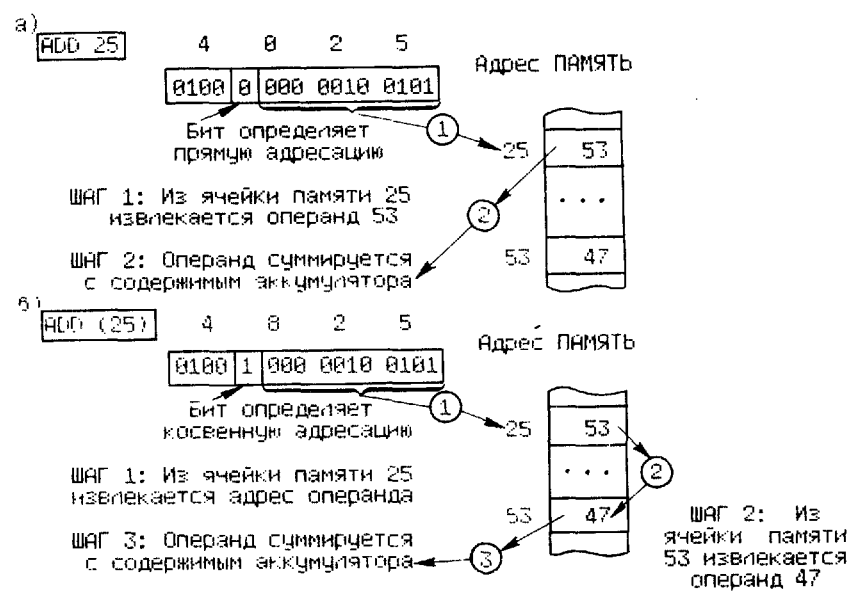


Рис 2.4. Способы адресации:
а — прямой; *б* — косвенный

щать не сам адрес операнда (результата или перехода), а его указатель, т. е. адрес ячейки памяти, в которой сохраняется адрес операнда (результата или перехода). Такое косвенное адресование упрощает построение циклических программ, организацию работы с подпрограммами, а также создает условия для расширения адресуемого пространства (косвенное адресование 16-битовых ячеек базовой ЭВМ позволяет ей иметь память объемом до $2^{16}=65\,536$ слов).

Для указания вида адресации в командах используется бит с номером 11 (рис. 2.3, *а*), в который при прямой адресации следует записывать 0, а при косвенной — 1. В мнемонических изображениях команд для указания косвенной адресации операнд помещается в скобки. Так, на рис. 2.4 команда ADD 25 или 4025 указывает, что из ячейки 25 должно быть взято число (53), которое нужно сложить с содержимым аккумулятора. Команда же ADD (25) или 4825 указывает, что из ячейки 25 должен быть взят адрес ячейки (53), в которой хранится число (47), которое и нужно сложить с содержимым аккумулятора. («А это веселая птица-синица, которая ловко ворует пшеницу, которая в темном чулане хранится в доме, который построил Джек.» Похоже?)

2.3. Арифметические операции

В этом параграфе рассматриваются основные способы записи чисел в базовой ЭВМ, арифметические операции, выполняемые с этими числами, и команды, реализующие такие операции. С рядом этих команд (CLA, ADD), командой пересылки (MOV) и командой останова (HLT) мы уже ознакомились.

Целые двоичные числа без знака. Такие числа можно использовать для представления нуля и целых положительных чисел. При размещении этих чисел в одном 16-битовом слове (см. рис. 1.1) они могут изменяться от $(0000\ 0000\ 0000\ 0000)_2=(0000)_{16}=0$ до $(1111\ 1111\ 1111\ 1111)_2=(FFFF)_{16}=$ $=2^{16}-1=65535$. Подобные числа (так же, как и рассмотренные ниже целые двоичные числа со знаком) относятся к числам с фиксированной запятой, поскольку в них положение запятой, разделяющей целую и дробную части числа, строго фиксировано после младшего бита (см. рис. 1.1 и 1.5, *а*).

Суммирование целых двоичных чисел без знака выполняется по тем же правилам, что и десятичное сложение, за исключением того, что перенос в следующий разряд производится после того, как сумма достигает 2 (1 + 1), а не 10. Рассмотрим несколько примеров сложения.

Пример 2.1. Сумма меньше 65536 (2^{16}):

```
      1    111  1          11          Переносы
+  0000 0110 0011 1010     1594
   0100 0100 1001 1011    17563
 ---------------------    -----
   0100 1010 1101 0101    19157
```

Пример 2.2. Сумма равна 65536:

```
   1                       111          Переносы
+  1000 0000 0000 0000     32768
   1000 0000 0000 0000     32768
 ---------------------    -----
 1 0000 0000 0000 0000      0    ?
```

Здесь в ответе должно получиться число $32768+32768=65536$, но поскольку разрядность слова составляет лишь 16 бит, то в нем сохраняется часть результата, т. е. 0. В данном и следующем примерах при сложении появляется единица переноса из старшего разряда, которая

оказывается за пределами представления числа (в несуществующем 17 разряде). Для сохранения этой единицы в ЭВМ используется специальный 1-битовый регистр переноса - С (см. параграф 2.1). Пользователь ЭВМ должен так составлять программы, чтобы при выполнении следующего за суммированием действия нужным образом учитывалось (использовалось или игнорировалось) содержимое регистра переноса.

Пример 2.3. Сумма больше чем 65536:

```
  1 1111 1111 1111 111        111          Переносы
+  1111 1111 1111 1111      +65535
   0111 1111 1111 1111       32767
 ---------------------      --------
 1 0111 1111 1111 1110       32766    ?
```

Здесь из-за ограниченной разрядности слова вместо 65535 + 32767 = 98302 опять (как и в примере 2.2) получен результат, уменьшенный на 65536 (2^{16}), и появляется перенос из старшего разряда.

В базовой ЭВМ для реализации рассмотренной операции используется команда ADD (СЛОЖЕНИЕ). Пример программы сложения двух чисел был рассмотрен в параграфах 2.1 и 2.2.

Целые двоичные положительные и отрицательные числа. Чтобы отличить отрицательные числа от положительных, в любых рассмотренных выше системах счисления можно использовать либо знак (такое представление чисел называется их прямым кодом), либо величину числа (одно из таких представлений чисел называется их дополнительным кодом).

Использование чисел со знаком (прямого кода представления чисел) усложняет структуру ЭВМ, так как операция сложения двух чисел, имеющих разные знаки, должна быть заменена на операцию вычитания меньшей величины из большей и присвоения результату знака большей величины. Поэтому в большинстве ЭВМ отрицательные числа представляют в виде так называемых дополнений, что при суммировании двух чисел с разными знаками позволяет заменить вычитание на обычное сложение.

Дополнением M n-разрядного целого числа K называется разность

$$M = b^n - K, \tag{2.1}$$

где b — основание системы счисления.

Так как количество различных целых n-разрядных чисел равно b^n (от 0 до $b^n - 1$), то половину этих чисел рассматривают как положительные (от 0 до $b^n/2 - 1$), а другую половину — как отрицательные (т. е. как дополнения к первой половине чисел). Полный же набор таких чисел называют числами, представленными в дополнительном коде (табл. 2.5).

Легко проверить правильность получения дополнений (кодов отрицательных чисел) в столбцах табл. 2.5. Для этого надо сложить число из нижней половины таблицы с его дополнением, расположенным над кодом 0. Результатом такого суммирования будут числа $10^5 = (1\,00\,000)_{10}$, $16^4 = (1\,0000)_{16}$ или $2^{16} = (1\,0\,000\,0000\,0000\,0000)_2$. В связи с тем, что разрядность чисел в табл. 2.5 ограничена 5, 4 и 16 разрядами соответственно, 1 в старшем разряде полученных сумм не учитывается, так как она «вываливается» за разрядную сетку. Следовательно, все суммы равны нулю, т. е. правильному результату сложения двух одинаковых по величине чисел, имеющих разные знаки.

Дополнения числа можно получить и без вычитания. Перепишем формулу (2.1) в виде

$$M = b^n - K = ((b^n - 1) - K) + 1. \tag{2.2}$$

Здесь число $b^n - 1$ состоит из n цифр $(b - 1)$, т. е. старших цифр используемой системы счисления (для чисел в табл. 2.5 это 99 999, FFFF или

Таблица 2.5

Дополнительные коды чисел

Прямой код ряда 5-разрядных десятичных чисел	Дополнительный код чисел		
	5-разрядных десятичных	4-разрядных шестнадцатеричных	16-разрядных двоичных
−50 000	50 000		—
−49 999	50 001		—
−49 998	50 002	—	—
...	...	...	...
−32 769	67 231		
−32 768	67 232	8000	1 000 0000 0000 0000
−32 767	67 233	8001	1 000 0000 0000 0001
...	...	...	...
−3	99 997	FFFD	1 111 1111 1111 1101
−2	99 998	FFFE	1 111 1111 1111 1110
−1	99 999	FFFF	1 111 1111 1111 1111
0	00 000	0000	0 000 0000 0000 0000
+1	00 001	0001	0 000 0000 0000 0001
+2	00 002	0002	0 000 0000 0000 0010
+3	00 003	0003	0 000 0000 0000 0011
...	...	...	...
+32 766	32 766	7FFE	0 111 1111 1111 1110
+32 767	32 767	7FFF	0 111 1111 1111 1111
+32 768	32 768		
...	...	...	...
+49 998	49 998		
+49 999	49 999	—	—

1 111 1111 1111 1111). Поэтому $(b^n - 1) - K$ можно получить путем образования дополнений до $b - 1$ для каждой из цифр числа K в отдельности, а искомое дополнение — суммированием этого числа с 1. Так как в двоичной системе дополнения цифр до 1 соответствуют их инверсным значениям (дополнение 0 равно 1, а дополнение 1 — 0), то можно дать простое правило для получения дополнения двоичных чисел.

1. Получить инверсию заданного числа (все его 0 заменить на 1, а все 1 — на 0):

```
0 000 0010 1100 0101        число
1 111 1101 0011 1010        инверсия числа
```

2. Образовать дополнительный код заданного числа путем добавления 1 к инверсии этого числа:

```
  1 111 1101 0011 1010      инверсия числа
+                    1      слагаемое 1
-------------------------
  1 111 1101 0011 1011      дополнительный код числа
```

Проверим правильность перевода:

```
  1 1 111 1111 1111 111     переносы
    0 000 0010 1100 0101    число
+   1 111 1101 0011 1011    дополнительный код числа
-------------------------
  1 0 000 0000 0000 0000    0
```

Так как перенос из старшего разряда не учитывается, то результат суммирования равен 0, что подтверждает правильность преобразования.

В заключение отметим еще одну особенность дополнительного кода двоичных чисел: их старший бит выполняет функцию знака числа, т. е. равен 0

для положительных чисел и 1 — для их дополнений (отрицательных чисел). Поэтому старший разряд ячеек памяти и регистров, которые могут хранить такие числа, обычно называют знаковым разрядом.

Сложение целых двоичных чисел со знаком. Выполняется оно по тем же правилам и с помощью той же команды ADD, что и для чисел без знака. Примеры сложения целых чисел X и Y в целях получения суммы Z приведены в табл. 2.6.

В двух случаях (втором и четвертом) ограниченная разрядность чисел приводит к искажению не только величины, но и знака результата. Исправить положение можно, добавляя к результату (или вычитая из него) число $2^{16}=65536$, но для этого надо переходить к размещению результата в двух словах.

Если слагаемые таковы, что вероятность получения искаженных результатов не равна нулю, то перед суммированием необходимо анализировать знаки слагаемых и результата. Разные знаки слагаемых или совпадение знаков слагаемых со знаком суммы свидетельствуют о том, что результат будет корректным. В противном случае вычисления должны быть прерваны и выдан сигнал ПЕРЕПОЛНЕНИЕ.

Таблица 2.6

Различные случаи сложения двоичных чисел со знаком

Случай	Слагаемые и результат	Комментарий
1	$X>0,\quad Y>0,\quad X+Y<2^{15}$ + 0 000 0110 0011 1010 X = +1594 0 100 0100 1001 1011 Y = +17563 0 100 1010 1101 0101 Z = +19157	Результат корректный
2	$X>0,\quad Y>0,\quad X+Y\geqslant 2^{15}$ + 0 100 0100 1001 1011 X = +17563 0 100 1010 1101 0101 Y = +19157 1 000 1111 0111 0000 Z = −28816	При сложении положительных чисел получен отрицательный результат ПЕРЕПОЛНЕНИЕ!
3	$X<0,\quad Y<0,\quad \lvert X\rvert+\lvert Y\rvert<2^{15}$ + 1 111 1001 1100 0110 X = −1594 1 011 1011 0110 0101 Y = −17563 1 1 011 0101 0010 1011 Z = −19157	Результат корректный. Перенос из старшего разряда не учитывается
4	$X<0,\quad Y<0,\quad \lvert X\rvert+\lvert Y\rvert>2^{15}$ + 1 011 1011 0110 0101 X = −17563 1 011 0101 0010 1011 Y = −19157 1 0 111 0000 1001 0000 Z = +28816	При сложении отрицательных чисел получен положительный результат ПЕРЕПОЛНЕНИЕ!
5	$X>0,\quad Y<0,\quad \lvert X\rvert>\lvert Y\rvert$ + 0 100 0100 1001 1011 X = +17563 1 111 1001 1100 0110 Y = −1594 1 0 011 1110 0110 0001 Z = +15969	Результат корректный. Перенос из старшего разряда не учитывается
6	$X>0,\quad Y<0,\quad \lvert X\rvert<\lvert Y\rvert$ + 0 000 0110 0011 1010 X = +1594 1 011 1011 0110 0101 Y = −17563 1 100 0001 1001 1111 Z = −15969	Результат корректный

Размещение целого двоичного числа в нескольких словах памяти. Используется оно для расширения диапазона представления таких чисел и, следовательно, для увеличения точности расчетов. Так, размещая число в двух 16-битовых словах, мы сможем работать с числами в диапазоне

$$\begin{array}{l} \text{от } (1\,000\,0000\,0000\,0000\,\underbrace{0000\,0000\,0000\,0000}_{\text{младшее слово}})_2 = -2147483648 = 2^{31} \\ \text{до } (\underbrace{0\,111\,1111\,1111\,1111}_{\text{старшее слово}}\,1111\,1111\,1111\,1111)_2 = 2147483647 = 2^{31} - 1 \end{array}$$

вместо диапазона от -32768 до $+32767$, в котором изменяются однословные двоичные числа со знаком (см. табл. 2.5).

Так как арифметико-логическое устройство базовой ЭВМ позволяет обрабатывать лишь 16-разрядные операнды, то для суммирования многословных чисел приходится выполнять определенную последовательность действий:

1) произвести сложение младших слов чисел, при этом в регистр переноса автоматически занесется 1 или 0 в зависимости от наличия или отсутствия переноса, возникающего при сложении старших разрядов суммируемых слов;

2) записать полученную сумму в младшее слово результата;

3) произвести сложение следующих по старшинству слов и содержимого регистра переноса, в конце операции в регистр переноса заносится перенос из старшего разряда суммируемых слов (1 или 0);

4) полученную сумму записать в соответствующее слово результата;

5) повторять пп. 3 и 4 до тех пор, пока не будут просуммированы все слова слагаемых.

Таблица 2.7

Программа суммирования двухсловных чисел

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
0	F200	CLA	Очистить аккумулятор (аккумулятор содержит 0)
1	400A	ADD A	Аккумулятор содержит младшее слово слагаемого X
2	400C	ADD C	Аккумулятор содержит сумму младших слов X и Y, а регистр переноса — перенос, возникающий при их сложении (в данном примере перенос равен 1)
3	300E	MOV E	Сумма младших слов X и Y помещается в младшее слово, отведенное для результата
4	F200	CLA	Очистить аккумулятор (подготовка к суммированию старших слов слагаемых)
5	5009	ADC 9	Аккумулятор содержит сумму старшего слова X и содержимого регистра переноса, в котором хранится перенос, полученный при сложении младших слов
6	400B	ADD B	Аккумулятор содержит сумму старших слов X и Y, а также переноса
7	300D	MOV D	Сумма старших слов X и Y помещается в старшее слово, отведенное для результата
8	F000	HLT	Останов ЭВМ (прекращение выборки команд)
9	00BC		Старшее слово слагаемого X
A	614E		Младшее слово слагаемого X
B	048F		Старшее слово слагаемого Y
C	F4EA		Младшее слово слагаемого Y
D	0000		Ячейка, отведенная для старшего слова результата; сюда запишется число 1356 $(054C)_{16}$
E	0000		Ячейка, отведенная для младшего слова результата; сюда запишется число 22072 $(5638)_{16}$

Таблица 2.8

Программа изменения знака числа

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
0	F200	CLA	Очистить аккумулятор (аккумулятор содержит 0)
1	4006	ADD 6	Аккумулятор содержит X
2	F400	CMA	Аккумулятор содержит инверсию X
3	F800	INC	К содержимому аккумулятора добавляется 1 и в нем образуется дополнительный код числа X
4	3007	MOV 7	Число —X помещается в ячейку, отведенную для Y
5	F000	HLT	Останов ЭВМ (прекращение выборки команд)
6	6844		Число X
7	0000		Ячейка, отведенная для Y; после исполнения программы сюда будет записано — X=97BC

В табл. 2.7 приведен пример программы сложения чисел X=12345678 $(BC614E)_{16}$ и Y=76543210 $(48FF4EA)_{16}$. В этой программе используется команда ADC B, обеспечивающая суммирование содержимых аккумулятора, ячейки с адресом B и регистра переноса, в котором хранится значение переноса (0 или 1), полученное при выполнении предыдущей операции. Результат суммирования сохраняется в аккумуляторе, а в регистр переноса заносится 1 или 0 в зависимости от наличия или отсутствия переноса из старшего разряда суммируемых слов.

Увеличение на единицу (INCrement) и уменьшение на единицу (DECrement). По команде INC к содержимому аккумулятора прибавляется единица, а по команде DEC — единица вычитается. Если при этом появляется перенос из старшего разряда, то в регистр переноса (C) заносится 1, в противном случае — 0.

Изменение знака числа. Выше уже говорилось, что для изменения знака числа, записанного в дополнительном коде, необходимо его инвертировать, а затем прибавить единицу к младшему разряду. На ЭВМ такую операцию можно выполнить с помощью команд CMA (инверсия содержимого аккумулятора) и INC.

В табл. 2.8 приведен пример программы для получения Y = —X.

Вычитание (X — Y). Оно может выполняться путем сложения уменьшаемого X и дополнительного кода вычитаемого Y. Однако это требует записи и исполнения нескольких команд (CLA, ADD Y, CMA, INC, ADD X). Для сокращения программы и времени выполнения вычитания в базовой ЭВМ предусмотрена команда вычитания SUB Y (CLA, ADD X, SUB Y), которая реализует те же действия за меньшее время.

При выполнении вычитания (так же, как и при выполнении сложения) возможно переполнение. Признаком возникновения переполнения является получение положительного результата при X < 0 и Y > 0 или отрицательного результата при X > 0 и Y < 0 (см. табл. 2.6).

Умножение и деление. В базовой ЭВМ нет команд для осуществления умножения и деления (АЛУ не выполняет этих операций). Поэтому пользователь должен организовывать получение произведения или частного программным путем.

При двоичном умножении частичное произведение сдвигается на один разряд влево для каждого последующего разряда множителя. Если множитель равен 0, то частичное произведение равно 0; если множитель равен 1, то частичное произведение равно множимому.

В качестве примера рассмотрим получение произведений $5\times5=25$, $5\times10=50$ и $5\times3=15$:

```
  101     (5)          101     (5)          101     (5)
×  11     (3)        × 101     (5)        × 1010    (10)
-----                -----                -------
  101                  101                   000
 101                  000                   101
-----                101                   000
 1111     (15)       -----                 101
                     11001    (25)        -------
                                          110010    (50)
```

2.4. Управление вычислительным процессом, сдвиги и логические операции

Почти все программы ЭВМ должны обладать способностью проверять исходные данные, промежуточные и окончательные результаты вычислений, чтобы на основании результатов проверки изменять нужным образом последовательность выполнения операций. Например, следует должным образом отреагировать на ситуацию, когда при сложении положительных чисел возникает отрицательный результат (см. параграф 2.3).

Подобная задача решается в базовой ЭВМ при помощи команд перехода (BCS, BPL, BMI, BEQ, BR), команд ПРИРАЩЕНИЕ, и ПРОПУСК (ISZ), ОСТАНОВ (HLT). Все эти команды (кроме HLT) являются адресными, т. е. в них указывается адрес той ячейки памяти, из которой должна быть выбрана следующая команда при выполнении того или иного условия. Если же условие не выполняется, то должна исполняться команда, расположенная вслед за данной командой управления. Как и в других адресных командах, здесь можно использовать косвенную адресацию, т. е. указывать адрес ячейки, из которой и будет выбираться нужный адрес перехода.

Разветвления в программах организуются с помощью команд перехода. Эти команды не изменяют состояния аккумулятора и регистра переноса. Они могут лишь изменить содержимое счетчика команд, поместив туда адрес, определяемый адресной частью команды.

BCS M (ПЕРЕХОД, ЕСЛИ ПЕРЕНОС). Переход к команде, расположенной в ячейке с адресом M, если содержимое регистра переноса равно 1.

BPL M (ПЕРЕХОД, ЕСЛИ ПЛЮС). Переход к команде, расположенной в ячейке с адресом M, если содержимое аккумулятора больше или равно нулю, т. е. в его старшем бите содержится 0.

BMI M (ПЕРЕХОД, ЕСЛИ МИНУС). Переход к команде, расположенной в ячейке с адресом M, если содержимое аккумулятора меньше нуля, т. е. в его старшем разряде содержится 1.

BEQ M (ПЕРЕХОД, ЕСЛИ НУЛЬ). Переход к команде, расположенной в ячейке с адресом M, если содержимое аккумулятора равно нулю т. е. во всех его разрядах содержится 0.

Если условие, указанное в перечисленных командах, не выполняется, то переход не может быть осуществлен и исполняется команда, расположенная за командой условного перехода.

BR M (БЕЗУСЛОВНЫЙ ПЕРЕХОД). Переход к команде, расположенной в ячейке с адресом M, осуществляемый при любых значениях аккумулятора, регистра переноса и других регистров ЭВМ.

Пример 2.4. Получить модуль числа X, содержащегося в ячейке памяти с адресом 100. Результат поместить в ячейку с адресом 200. Программа получения модуля приведена в табл. 2.9.

Таблица 2.9

Программа получения модуля

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
0	F200	CLA	Аккумулятор содержит 0
1	4100	ADD 100	Аккумулятор содержит аргумент X
2	9005	BPL 5	Если $X \geqslant 0$, то перейти к команде, расположенной в ячейке с адресом 5
3	F400	CMA	Получение дополнительного кода числа X
4	F800	INC	
5	3200	MOV 200	Запись $\lvert X \rvert$ в ячейку с адресом 200
6	F000	HLT	Останов ЭВМ

Циклические программы используются в тех случаях, когда требуется несколько раз выполнить набор одинаковых действий над некоторым изменяющимся составом данных.

Базовая ЭВМ обладает рядом средств для упрощения организации циклических программ. Целесообразность введения этих средств удобнее всего рассмотреть на примерах.

Сначала рассмотрим получение произведения $Z = 50Y$. Так как в системе команд базовой ЭВМ нет команды умножения, то воспользуемся простейшим способом: будем 50 раз складывать значение Y. При этом линейная программа, содержащая 50 команд ADD Y, вряд ли займет первое место в конкурсе программ для умножения. Ее, наверное, опередит программа, приведенная в табл. 2.10.

Существуют ли пути упрощения этой программы? Да.

Например, вместо подсчета и проверки количества выполненных циклов можно подсчитывать и проверять количество циклов, оставшихся до завершения работы программы, т. е. уменьшать в цикле значение M до тех пор, пока оно не станет равным нулю. Тогда отпадает необходимость в использовании ячейки 8 и команды вычитания (SUB 7) — команды с 15 по 19 заменяются последовательностью команд: ADD 7, DEC, MOV 7, BEQ 1A, BR 10. Если же в ячейку 7 поместить отрицательное значение множителя $-50 = (FFCE)_{16}$, то команды с 15 по 19 можно заменить последовательностью команд: ADD 7, INC, MOV 7, BMI 10, HLT, т. е. освободить еще и ячейку 1A.

Большего упрощения программы можно добиться при накоплении результата непосредственно в аккумуляторе, а не в ячейке 6. Тогда команды с 10 по 13 можно было бы заменить всего одной командой ADD 5. Однако в обсуждаемом варианте программы (см. табл. 2.10) аккумулятор пришлось использовать еще и для изменения числа циклов, что привело к необходимости сохранения промежуточного результата Z в памяти ЭВМ. Следовательно, для упрощения циклических программ целесообразно иметь в составе команд базовой ЭВМ команду, позволяющую организовать счетчик циклов без затрагивания аккумулятора. Такой командой является ISZ.

Команда ISZ (ПРИРАЩЕНИЕ И ПРОПУСК) служит для увеличения на 1 содержимого адресуемой ячейки памяти и перехода к одному из двух путей продолжения программы в зависимости от знака этого содержимого. Так, при каждом выполнении команды ISZ M, расположенной по адресу A, к содержимому ячейки с адресом M добавляется 1, и если результат меньше нуля, то выполняется команда, следующая за ISZ M (команда, расположен-

Таблица 2.10

Первый вариант программы для получения Z = 50Y

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
5	0078	Y	Множимое (здесь десятичное значение 120)
6	0000	Z	Ячейка, отведенная для накопления результата. В ней поочередно будут храниться значения Y, 2Y, 3Y,... После 50 суммирований в ней будет содержаться искомый результат 50 Y
7	0032	M	Множитель $50 = (32)_{16}$
8	0000	C	Ячейка, используемая для накопления числа выполненных циклов,
	...		
10	F200	CLA	
11	4006	ADD 6	К промежуточному результату, находящемуся в ячейке 6, добавляется еще одно значение множимого Y
12	4005	ADD 5	
13	3006	MOV 6	
14	F200	CLA	
15	4008	ADD 8	Содержимое счетчика циклов увеличивается на 1, а его копия пока сохраняется в аккумуляторе
16	F800	INC	
17	3008	MOV 8	
18	6007	SUB 7	Если содержимое счетчика циклов меньше значения множителя, то выполняется переход к командам, осуществляющим очередное суммирование Y с промежуточным значением Z. В противном случае выполняется следующая команда (команда HLT)
19	A010	BMI 10	
1A	F000	HLT	Останов ЭВМ. В ячейке 6 содержится результат

ная по адресу A + 1); в противном случае эта команда пропускается, т. е. выполняется команда, расположенная по адресу A + 2. Очень ценным свойством команды ISZ M является то, что она не портит содержимого аккумулятора и регистра переноса (увеличение M производится без использования этих регистров).

В табл. 2.11 приведен вариант программы получения Z = 50Y, в котором использованы все обсужденные выше возможности сокращения памяти и времени для накопления результата (время уменьшается за счет того, что в каждом из 50 циклов выполняются лишь три команды вместо десяти).

В качестве следующего примера рассмотрим получение в ячейке 005 суммы 32 элементов массива, размещенного в ячейках памяти с 010 по 02F.

В отличие от предыдущей задачи, где многократно суммировалось содержимое одной ячейки (Y), здесь надо суммировать содержимое разных ячеек. Естественно, нас не может устроить программа, содержащая последовательность из 32 команд сложения: 4010, 4011, ..., 402E, 402F. Очевидно, что требуется организовать циклический процесс, в котором при каждом прохождении цикла должен изменяться адрес суммируемого элемента массива (первый раз этот адрес должен быть равен 10, при втором прохождении цикла — 11, при третьем — 12 и т. д.). Как же изменять адрес суммируемого элемента?

В первых ЭВМ изменение адреса осуществлялось путем модификации адресной части команды (переадресации): после того как содержимое какой-либо ячейки памяти было использовано ЭВМ в качестве команды, последующие команды программы использовали это содержимое в качестве операнда (числа) и преобразовывали его нужным образом (например, наращивали

Таблица 2.11

Второй вариант программы для получения Z=50Y

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
5	0078	Y	Множимое
6	0000	Z	Ячейка, отведенная для результата
7	FFCE	M	Отрицательное значение множителя (—50)
10	F200	CLA	Очистка аккумулятора
11	4005	ADD 5	К содержимому аккумулятора добавляется Y
12	0007	ISZ 7	Содержимое M наращивается на 1, и если оно еще
13	C011	BR 11	меньше нуля, то выполняется команда BR 11. При M=0 команда BR 11 пропускается
14	3006	MOV 6	Результат 50 сложений Y записывается в ячейку 6
15	F000	HLT	Останов ЭВМ

на 1). Воспользуемся этим приемом для решения поставленной задачи (табл. 2.12).

Переадресация команд практически не используется в современных ЭВМ. Для сближения языка команд с алгоритмическими языками были разработаны специальные средства адресации (часть из них кратко рассматривалась в параграфе 2.2). Эти же средства позволяют создавать эффективные программы, помещаемые в постоянные запоминающие устройства, откуда можно лишь читать команды или данные, но нельзя их изменять (программы микроЭВМ, управляющих работой стиральных машин, некоторых устройств автомобиля, роботом и т. п.). Одно из таких средств — косвенная адресация (см. рис. 2.4).

При использовании [illegible] нужно выбрать в памяти ЭВМ какую-либо ячейку (например, 007), записать в нее адрес первого элемента суммируемого массива (адрес 010), заменить в программе табл. 2.12 команду

Таблица 2.12

Первый вариант программы суммирования элементов массива

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
5	0000		Ячейка, отведенная для накопления результата
6	FFE0		Отрицательное число элементов массива (—32)
10			Численные значения элементов массива
	...		
2F			
30	F200	CLA	Промежуточный результат (ячейка 5) суммируется
31	4005	ADD 5	с содержимым элемента массива, адрес которого рас-
32	4010	?	положен в адресной части команды, находящейся
33	3005	MOV 5	в ячейке 32 (сначала этот адрес равен 10, а затем он при каждом прохождении цикла увеличивается на 1 с помощью команд 34—37)
34	F200	CLA	Пересылка в аккумулятор команды, расположенной
35	4032	ADD 32	по адресу 32, добавление к ее содержимому 1 и запись
36	F800	INC	модифицированной команды на старое место (в ячей-
37	3032	MOV 32	ку 32)
38	0006	ISZ 6	Наращивание на 1 счетчика элементов массива и
39	C030	BR 30	переход к команде 30, пока его содержимое меньше нуля
3A	F000	HLT	Останов ЭВМ

Таблица 2.13

Второй вариант программы суммирования элементов массива

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
5	0000		Ячейка, отведенная для накопления результата
6	FFE0		Отрицательное число элементов массива (−32)
7	0010		Текущий адрес элемента массива (сначала адрес первого элемента массива 10)
10			Численные значения элементов массива
	...		
2F			
30	F200	CLA	Очистка аккумулятора
31	4807	ADD (7)	Суммирование очередного элемента массива
32	0007	ISZ 7	Текущий адрес элемента массива наращивается на 1
33	F100	NOP	Команда НЕТ ОПЕРАЦИИ
34	0006	ISZ 6	Наращивание на 1 счетчика элементов массива и переход к команде 31, пока его содержимое меньше нуля
35	C031	BR 31	
36	3005	MOV 5	Запись результата в ячейку 5
37	F000	HLT	Останов ЭВМ

4010 на команду 4807 (ячейка 32) и команды переадресации (CLA, ADD 32, INC, MOV 32) командами вычисления текущего адреса суммируемого элемента массива (CLA, ADD 7, INC, MOV 7). Скобки в команде ADD (7) и цифра 8 в ее числовом коде 4807 (единица в 11-м бите команды) информируют ЭВМ о том, что содержимое аккумулятора должно суммироваться не с содержимым ячейки 7, а с содержимым ячейки, адрес которой хранится в ячейке 7. Поэтому в ячейку 7 сначала помещается адрес первого элемента суммируемого массива, а затем (при каждом прохождении цикла) этот адрес увеличивается на 1.

В табл. 2.13 приведен укороченный вариант программы суммирования элементов массива, где кроме косвенной адресации используется и новое средство для изменения текущего адреса суммируемого элемента массива. Этим средством является команда ISZ 7, при выполнении которой содержимое ячейки 7 наращивается на 1. Но так как это содержимое (адрес элемента массива) — положительная величина, то после выполнения ISZ 7 будет пропущена следующая за ней команда. Поэтому в ячейку 33 помещена команда NOP (НЕТ ОПЕРАЦИИ), но можно было бы поместить любое число.

Наконец, рассмотрим еще одно средство, позволяющее упростить циклические программы базовой ЭВМ, — индексные ячейки (ячейки с адресами от 008 до 00F). Если произвести косвенное адресование какой-либо из этих ячеек, то сначала ее содержимое будет использовано в качестве адреса операнда, а затем оно автоматически увеличится на единицу. При прямом адресовании индексные ячейки работают как обычные ячейки (их содержимое может измениться лишь в случае записи информации в ячейку).

Указанное свойство индексных ячеек позволяет составить оптимальную программу для суммирования элементов массива (табл. 2.14).

[illegible] ных обеспечивается в базовой ЭВМ командами логического умножения (AND), циклического сдвига влево (ROL) и вправо (ROR), а также командами инвертирования (CMC) и очистки (CLC) регистра переноса. Эти команды часто используются для упаковки информации в памяти ЭВМ (например, размещения в одном слове нескольких

Таблица 2.14

Третий вариант программы суммирования элементов массива

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
5	0000		Ячейка, отведенная для накопления результата
6	FFE0		Отрицательное число элементов массива (—32)
F	0010		Текущий адрес элемента массива
10			Численные значения элементов массива
2Г	...		
30	Г200	CLA	Очистка аккумулятора
31	480F	ADD (F)	Суммирование очередного элемента массива. Так как сначала в индексную ячейку F помещен адрес первого элемента массива (10), то после первого выполнения данной команды содержимое ячейки F увеличится на 1 и будет указывать на второй элемент массива, после второго выполнения — на третий элемент массива и т. д.
32	0006	ISZ 6	Наращивание на 1 счетчика элементов массива и переход к команде 31, пока его содержимое меньше нуля
33	C031	BR 31	
34	3005	MOV 5	Запись результата в ячейку 5
35	F000	HLT	Останов ЭВМ

небольших чисел или кодов) и ее распаковки, обработки текстовой информации, умножения (деления) на 2, 4, 8 ... и т. п.

Команда AND M выполняет над каждым разрядом содержимого аккумулятора и содержимым ячейки памяти с адресом M булеву операцию «И»

Таблица 2.15

Программа определения количества нечетных элементов массива

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
5	0000		Ячейка, отведенная для накопления результата
6	0001		Маска для выявления нечетных элементов массива. При ее логическом умножении на нечетное число результат будет равен 1 (иначе —0)
7	FFE0		Отрицательное число элементов массива (—32)
8	0010		Текущий адрес элемента массива (индексная ячейка)
10			Численные значения элементов массива
21	...		
30	F200	CLA	Очистка аккумулятора
31	4808	ADD (8)	Аккумулятор содержит очередной элемент массива
32	1006	AND 6	Выделяется младший бит аккумулятора (элемента массива) и добавляется к сумме числа нечетных элементов массива. Если младший бит равен 1 (нечетный элемент), то сумма увеличивается на эту единицу
33	4005	ADD 5	
34	3005	MOV 5	
35	0007	ISZ 7	Модификация содержимого счетчика элементов массива и переход по адресу 30, пока это содержимое меньше нуля
36	C030	BR 30	
37	F000	HLT	Останов ЭВМ

Рис. 2.5. Циклические сдвиги:
а — влево; *б* — вправо

Результат выполнения команды для каждой пары битов операндов равен единице только тогда, когда оба бита равны единице; в остальных случаях бит результата равен нулю. Так как аналогичный результат получается и при поразрядном умножении битов ($0\times0=0$, $0\times1=0$, $1\times0=0$ и $1\times1=1$), то команда называется логическим умножением. Эти свойства позволяют выделять или очищать определенные биты слова.

В качестве последнего примера упрощения циклических программ рассмотрим получение количества нечетных элементов массива, расположенного в ячейках с адресами от 010 до 02F. Программа для решения этой задачи приведена в табл. 2.15.

Таблица 2.16

Программа выделения знака числа

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
7	FFE0		Отрицательное число элементов массивов X и H
8	0010		Текущий адрес элемента массива X
9	0030		Текущий адрес элемента массива H
10			Численные значения массива X
	...		
2F			
30			Численные значения массива H
	...		
4F			
50	F200	CLA	Очистка аккумулятора
51	4808	ADD (8)	В аккумуляторе очередной элемент массива X
52	F600	ROL	Сдвиг знакового разряда X(I) в регистр переноса
53	F200	CLA	Очистка аккумулятора
54	F600	ROL	Сдвиг содержимого регистра переноса в младший
55	3809	MOV (9)	разряд аккумулятора и запись полученного значения в H(I)
56	0007	ISZ 7	Модификация содержимого счетчика элементов
57	C050	BR 50	массивов и переход по адресу 50, пока это содержимое меньше нуля
58	F000	HLT	Останов ЭВМ

Команды ROL (ЦИКЛИЧЕСКИЙ СДВИГ ВЛЕВО НА 1 РАЗРЯД) и ROR (ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО НА 1 РАЗРЯД) замыкают аккумулятор и регистр переноса в кольцо и сдвигают все биты кольца на один разряд влево или вправо (рис. 2.5). Сдвигами битов числа влево или вправо можно реализовать операции умножения или деления на 2 (один сдвиг), 4 (два сдвига) и т. д. Действительно, после сдвига числа

$$(0000\ 0000\ 0011\ 1010)_2 = 58$$

на один разряд влево будет получено число

$$(0000\ 0000\ 0111\ 0100)_2 = 116,$$

а после еще одного такого сдвига — число

$$(0000\ 0000\ 1110\ 1000)_2 = 232$$

(здесь предполагалось, что регистр переноса был очищен перед сдвигами).

Рассмотрим еще один пример упрощения программы за счет использования команд сдвигов: проанализировать элементы массива X (ячейки 010 – 02F) и при отрицательном X(I) присвоить H(I)-му элементу массива H значение 1, а при положительном X(I) — значение 0 (табл. 2.16).

В некоторых приложениях возникает необходимость инвертирования или очистки определенного бита слова. Для этого, например, можно:

поместить слово в аккумулятор;

сдвинуть слово так, чтобы нужный бит попал в регистр переноса;

выполнить команду CMC («Инвертирование регистра переноса») или CLC («Очистка регистра переноса»);

сдвинуть биты слова на старое место.

2.5. Подпрограммы и параметры

Достаточно часто встречается ситуация, когда отдельные части программы должны выполнять одни и те же действия по обработке данных (например, вычисление тригонометрической функции). В подобных случаях повторяющиеся части программы выделяют в подпрограмму, а в соответствующие места программы заносят лишь команды обращения к этой подпрограмме. В базовой ЭВМ для этой цели используется команда JSR («Обращение к подпрограмме»).

На рис. 2.6 показана часть основной программы, содержащая две команды JSR 300, с помощью которых осуществляется переход к выполнению команд подпрограммы.

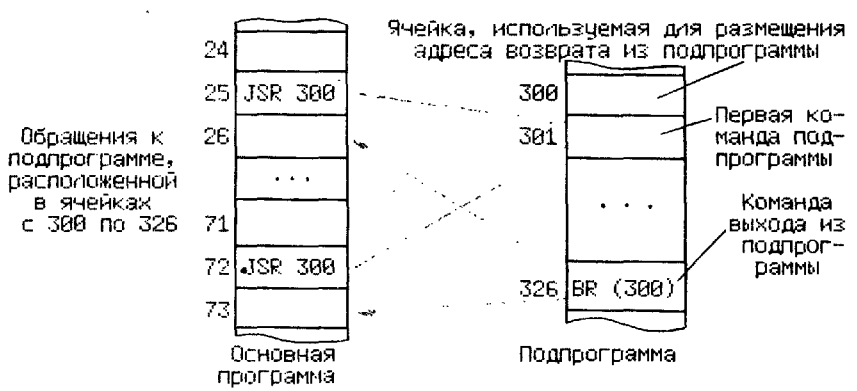


Рис. 2.6. Обращение к подпрограмме из основной программы

По команде JSR 300, расположенной в ячейке 25, выполняются запись числа 25+1=26 в ячейку с адресом 300 и переход к первой исполняемой команде подпрограммы, расположенной в ячейке 301. Далее начинается процесс выполнения команд подпрограммы, который завершается на команде BR (300), расположенной в ячейке 326. Это команда безусловного перехода с косвенной адресацией. Она предписывает ЭВМ выполнить переход к команде, расположенной по адресу, сохраняемому в ячейке 300. Так как в эту ячейку ранее было занесено число 26, то будет исполняться команда, записанная в ячейке 26, т. е. следующая за обращением к подпрограмме.

По команде JSR 300, расположенной в ячейке 72, осуществляется запись числа 72+1=73 в ячейку с адресом 300 и начинается выполнение подпрограммы (с ячейки 301). При исполнении последней команды подпрограммы BR (300) будет осуществлен переход к команде, расположенной по адресу 73, т. е. адресу, хранимому в ячейке 300.

Таким образом, при оформлении подпрограммы перед первой ее командой следует разместить ячейку, в которую будет пересылаться адрес возврата из подпрограммы. В обращении к подпрограмме указывается адрес именно этой ячейки, и команда JSR M выполняет следующие действия:

1) помещает в ячейку M адрес следующей за JSR M команды, т. е. команды, которая должна выполняться вслед за командами подпрограммы;

2) передает управление команде, расположенной в ячейке с адресом M+1.

Таблица 2.17

Первый вариант программы получения удвоенного модуля трех чисел

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
		Основная программа	
10 11 12 13	F200 4058 2030 3074	CLA ADD 58 JSR 30 MOV 74	Запись в аккумулятор значения X, обращение к подпрограмме получения удвоенного модуля (запись в ячейку 30 адреса 13 и переход к команде, расположенной по адресу 31), пересылка результата в ячейку для хранения 2 {X}
14 15 16 17	F200 4071 2030 3077	CLA ADD 63 JSR 30 MOV 77	Те же действия, что и в командах 10–13, но над аргументом Y. При обращении к подпрограмме в ячейку 30 пересылается адрес возврата 17
18 19 1A 1B	F200 4063 2030 3082	CLA ADD 71 JSR 30 MOV 82	Те же действия, что и в командах 10–13, но над аргументом Z. При обращении к подпрограмме в ячейку 30 пересылается адрес возврата 1B
1C	F000	HLT	Останов ЭВМ
		Подпрограмма получения удвоенного модуля	
30 31	0000 9034	 BPL 34	Ячейка для размещения адреса возврата Первая команда, проверяет знак числа в аккумуляторе, если он положительный, то передает управление командам удвоения содержимого аккумулятора
32 33	F400 F800	CMA INC	Получение дополнительного кода
34 35	F600 C830	ROL BR (30)	Удвоение содержимого аккумулятора Выход из подпрограммы (переход к выполнению команды, адрес которой хранится в ячейке 30)

Таблица 2.18

Второй вариант программы получения удвоенного модуля трех чисел

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
		Основная программа	
10 11 12	2030 0058 0074	JSR 30	Обращение к подпрограмме, за которым располагаются данные: адрес X и ячейки для размещения 2 |X|. В ячейку 30 передается адрес 11
13 14 15	2030 0063 0077	JSR 30	Обращение к подпрограмме, за которым располагаются данные: адрес Y и ячейки для размещения 2 |Y|. В ячейку 30 передается адрес 14
16 17 18	2030 0071 0082	JSR 30	Обращение к подпрограмме, за которым располагаются данные: адрес Z и ячейки для размещения 2 |Z|. В ячейку 30 передается адрес 17
19	F000	HLT	Останов ЭВМ
		Подпрограмма получения удвоенного модуля	
30	0000		Ячейка для размещения адресов аргумента, результата и возврата из подпрограммы
31 32 33 34 35	F200 4830 3035 0030 0000	CLA ADD (30) MOV 35 ISZ 30	В ячейку 35 пересылается значение переменной, на которую указывает содержимое ячейки 30 (при обращении к подпрограмме это содержимое является адресом аргумента), а затем адрес в ячейке 30 наращивается на 1 (теперь он указывает на адрес результата)
36 37 38 39 3A	F200 4830 303A 0030 0000	CLA ADD (30) MOV 3A ISZ 30	В ячейку 3A пересылается адрес результата, и адрес в ячейке 30 увеличивается на 1 (теперь он указывает на адрес той команды основной программы, которая расположена вслед за обращением к данной подпрограмме)
3B 3C 3D 3E 3F 40 41	F200 4835 9040 F400 F800 F600 383A	CLA ADD (35) BPL 40 CMA INC ROL MOV (3A)	Получение удвоенного модуля числа, адрес которого указывается содержимым ячейки 35, и пересылка результата по адресу, на который указывает содержимое ячейки 3A
42	C830	BR (30)	Выход из подпрограммы

Примечание. Возможность размещения адресов аргумента и результата в ячейках, расположенных за командой ISZ, поясняла а табл. 2.13.

Последней исполняемой командой подпрограммы должна быть команда выхода, т. е. команда BR (M), по которой осуществляется переход к команде с адресом, хранимым в ячейке M.

Пример 2.5. Получить удвоенный модуль чисел X, Y и Z, хранящихся в ячейках памяти 58, 63 и 71 соответственно. Результаты (2|X|, 2|Y|, 2|Z|) поместить в ячейки 74, 77 и 82.

Для решения этой задачи можно взять за основу программу примера 2.4, оформив ее фрагмент в виде подпрограммы (табл. 2.17). В программе табл. 2.17 в подпрограмму передается всего одна величина: в аккумулятор помещается число, модуль которого нужно удвоить. Результат вычисления (тоже одна величина) возвращается в основную программу через тот же аккумулятор. А как же передать и (или) получить два или большее число параметров?

В микроЭВМ, имеющих несколько аккумуляторов или, лучше сказать, регистров общего назначения, можно передавать в подпрограмму (или

получать из нее) большее число параметров. Но и число регистров общего назначения также ограничено. Поэтому во всех микроЭВМ (как и в базовой ЭВМ) есть возможность обмена параметрами через ячейки памяти.

В табл. 2.18 приведен второй вариант программы для решения примера 2.7. Здесь использован один из возможных способов передачи параметров: размещение параметров непосредственно за командой JSR. Адрес возврата, записываемый командой JSR, в действительности является адресом первого параметра. В подпрограмме приходится производить наращивание этого адреса для выборки остальных параметров и получения действительного адреса возврата — адреса команды, расположенной за последним из параметров. В качестве параметров можно использовать константы или переменные (адреса). В последнем случае несколько усложняется обработка этих параметров (см. табл. 2.18).

Возможно и размещение параметров в специальной области параметров с передачей в подпрограмму начального адреса этой области. Основная программа может использовать эту область для нескольких подпрограмм, что позволит сократить объем памяти.

2.6. Выполнение машинных команд

Время, затрачиваемое на получение работоспособной программы ЭВМ, резко сокращается, если программист хорошо представляет, как выполняются отдельные машинные команды, и использует эти знания при отладке программы. Подробные сведения о процессах, происходящих в базовой ЭВМ во время выполнения отдельных команд, приведены в параграфах 4.2–4.4, а здесь кратко рассмотрим лишь те из них, знание которых позволяет быстрее локализовать ошибки при исполнении отлаживаемой программы.

В процессе исполнения машинных команд устройство управления ЭВМ производит анализ и пересылку команды, отдельных ее частей (кода операции, признака адресации и адреса) или операнда из одного регистра машины в другой ее регистр, АЛУ, память или устройство ввода-вывода.

Эти действия (микрооперации) протекают в определенной временно́й последовательности и скоординированы между собой. Для создания временно́й последовательности (см. рис. 1.4) используется генератор тактовых импульсов (импульсов с частотой в несколько МГц).

Цикл команды. Для реализации одной команды требуется выполнить определенное количество микрокоманд, каждая из которых инициирует одновременное осуществление одной или нескольких микроопераций за время одного рабочего такта ЭВМ. Общее число тактовых импульсов (микрокоманд), требуемых для выполнения команды, определяет время ее выполнения, называемое циклом команды (рис. 2.7). Цикл команды обычно включает один или несколько машинных циклов.

Устройство управления базовой ЭВМ может находиться в четырех возможных состояниях: выборки команды, выборки адреса, исполнения и прерывания. Длительность каждого из этих четырех состояний определяет время выполнения соответствующего машинного цикла. Каждый машинный цикл предназначен для определенной цели

Выборка команды. В данном машинном цикле выполняются чтение команды из памяти, ее частичное декодирование и иногда исполнение (для безадресных команд и команд ввода-вывода, являющихся [illegible]):

1) содержимое ячейки памяти, указываемой счетчиком команд, читается из памяти в регистр данных (см. рис. 2.2, *а*, *б*);

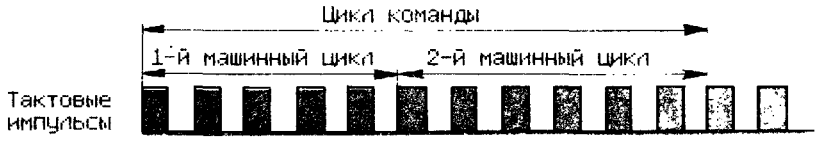


Рис. 2.7. Типичный цикл команды

2) содержимое счетчика команд увеличивается на единицу (см. рис. 2.2, *б*, *в*);

3) содержимое регистра данных пересылается в регистр команд (см. рис. 2.2, *г*), код операции команды частично декодируется для выяснения типа команды (адресная, безадресная или ввода-вывода), анализируется бит признака адресации и происходит подготовка цепей, необходимых для выполнения команды;

4) если выбрана адресная команда, то осуществляется переход к микрокомандам следующего машинного цикла; в противном случае выполняются действия по завершению одноцикловой команды.

На рис. 2.8, *а*, *б* показаны действия, выполняемые в конце цикла выборки для одноцикловой команды СМА, инвертирующей содержимое аккумулятора (5).

Выборка адреса. Это состояние (этот машинный цикл) следует за выборкой команды для адресных команд [illegible] (бит вида адресации равен единице). Состояние используется для чтения адреса операнда (адреса результата или перехода) из памяти и состоит из следующих шагов:

1) адресная часть команды пересылается из регистра данных (где пока еще сохраняется копия команды) в регистр адреса (рис. 2.9, *а*);

2) содержимое ячейки памяти, указываемой регистром адреса, читается в регистр данных (рис. 2.9, *б*); теперь в этом регистре находится адрес операнда (адрес результата или перехода), который будет использоваться при выполнении команды (в цикле исполнения).

Для иллюстрации действий, осуществляемых в этом машинном цикле, выбрана команда ADD (21), расположенная в ячейке 43. После выборки в регистре данных и регистре команд хранится сама команда (4821): четыре старших разряда — код операции $(0100)_2$, затем признак адресации $(1)_2$ и, наконец, адрес $(000\,0010\,0001)_2 = (21)_{16}$. В счетчике команд записан адрес следующей команды (44), в аккумуляторе результат предыдущих операций (7) и в 11-разрядном регистре адреса адрес исполняемой коман-

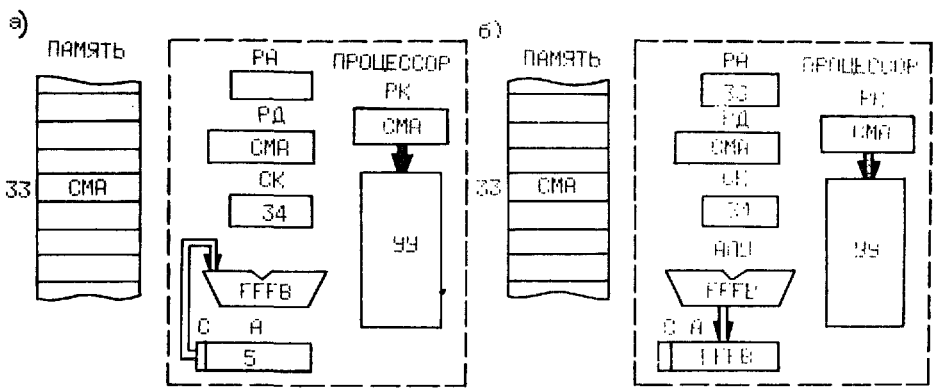


Рис. 2.8. Исполнение команды СМА в конце цикла выборки команды

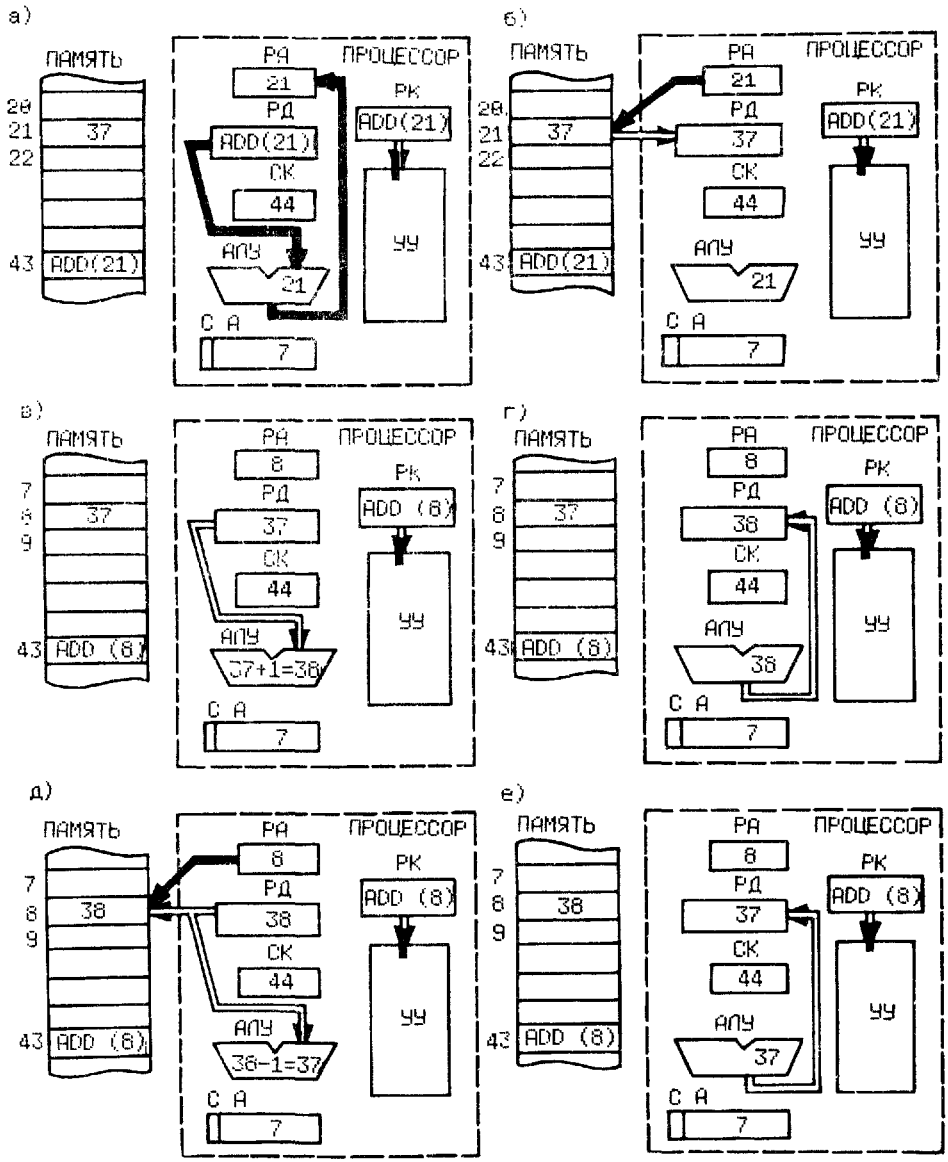


Рис. 2.9. Цикл выборки адреса

ды (43). В начале цикла выборки адреса содержимое регистра адреса заменяется на 11 младших разрядов регистра данных (рис. 2.9, *а*), т. е. на адрес адреса операнда, а затем в регистр данных считывается сам адрес операнда (рис. 2.9, *б*).

Если косвенно адресуется одна из индексных ячеек (ячеек с номерами от 008 до 00F), то цикл выборки адреса операнда (результата) продолжается:

3) содержимое регистра данных увеличивается на единицу (рис. 2.9, *в*, *г*);

4) измененное содержимое регистра данных пересылается в ячейку памяти по адресу, указываемому регистром адреса (рис. 2.9, *д*);

5) содержимое регистра данных уменьшается на 1 (рис. 2.9, *д*, *е*).

После этой операции в регистре данных восстанавливается значение адреса, находившегося в индексной ячейке до выполнения шага 3. Содержимое же индексной ячейки увеличилось на единицу, и при следующем обращении к ней будет выбран новый адрес операнда (результата)

Для иллюстрации шагов 3, 4 и 5 выбрана команда ADD (8), также расположенная в ячейке 43. Поэтому два первых шага по выборке адреса операнда (результата) будут очень похожи на аналогичные действия при выполнении команды ADD (21), которые показаны на рис. 2.9, *а*, *б*. Различие лишь в том, что в регистрах данных и команд первоначально сохранялась команда ADD (8) вместо ADD (21) и, следовательно, регистр адреса указывает на ячейку 8 (а не на ячейку 21).

Исполнение. Последовательность действий, выполняемых в этом цикле, определяется типом выполняемой адресной команды.

А. Для команд, при выполнении которых требуется выборка операнда из памяти ЭВМ (AND, ADD, ADC, SUB, ISZ), состояние исполнения используется для чтения операнда в регистр данных и выполнения операции, указываемой кодом операции команды.

Пример цикла исполнения для команды ADD 21 подробно рассмотрен на рис. 2.2, *д* — 2.2, *з*. И хотя на рис. 2.2 иллюстрировалось выполнение команды сложения с прямой адресацией, оба цикла (выборки и исполнения) будут совершенно такими же и при сложении с косвенной адресацией. В последнем случае между этими циклами будет выполнен цикл выборки адреса операнда, и этот адрес, так же, как и при прямой адресации, будет помещен в регистр данных.

Б. По команде пересылки (MOV) в этом машинном цикле производится запись содержимого аккумулятора в ячейку памяти с адресом, расположенным в регистре данных, для чего содержимое регистра данных пересылается в регистр адреса, а содержимое аккумулятора — в регистр данных и далее в ячейку памяти, указываемую регистром адреса

В. При исполнении команд переходов (BCS, BPL, BMI, BEQ) производятся проверка соответствующего условия (1 в регистре переноса, 0 в знаковом разряде аккумулятора и т. п.) и пересылка адреса из регистра данных в счетчик команд при выполнении этого условия. Если исполняется команда безусловного перехода (BR), то пересылка адреса перехода в счетчик команд осуществляется без какой-либо проверки.

Следовательно, при выполнении условия, определяемого кодом операции команды, или при выполнении команды BR следующей будет выбираться команда из ячейки памяти с адресом, расположенным в регистре данных (адресом, расположенным в исполняемой команде или выбранным из ячейки, на которую указывает этот адрес). В противном случае будет выбрана команда, расположенная вслед за исполняемой.

Г. Для команды обращения к подпрограмме (JSR) во время этого машинного цикла осуществляются пересылка содержимого счетчика команд в ячейку памяти, адрес которой содержится в регистре данных, и занесения в счетчик команд увеличенного на единицу содержимого регистра данных.

Эти действия иллюстрируются на примере выполнения команды JSR 30 (рис. 2.10). Здесь в качестве регистра для временного хранения адреса первой команды подпрограммы служит регистр команд, содержимое которого после декодирования команды уже не используется устройством управления.

Сначала адресная часть команды пересылается из регистра данных в регистр адреса (рис. 2.10, *а*). Затем этот же адрес наращивается на единицу (формируется адрес первой команды подпрограммы) и пересылается на

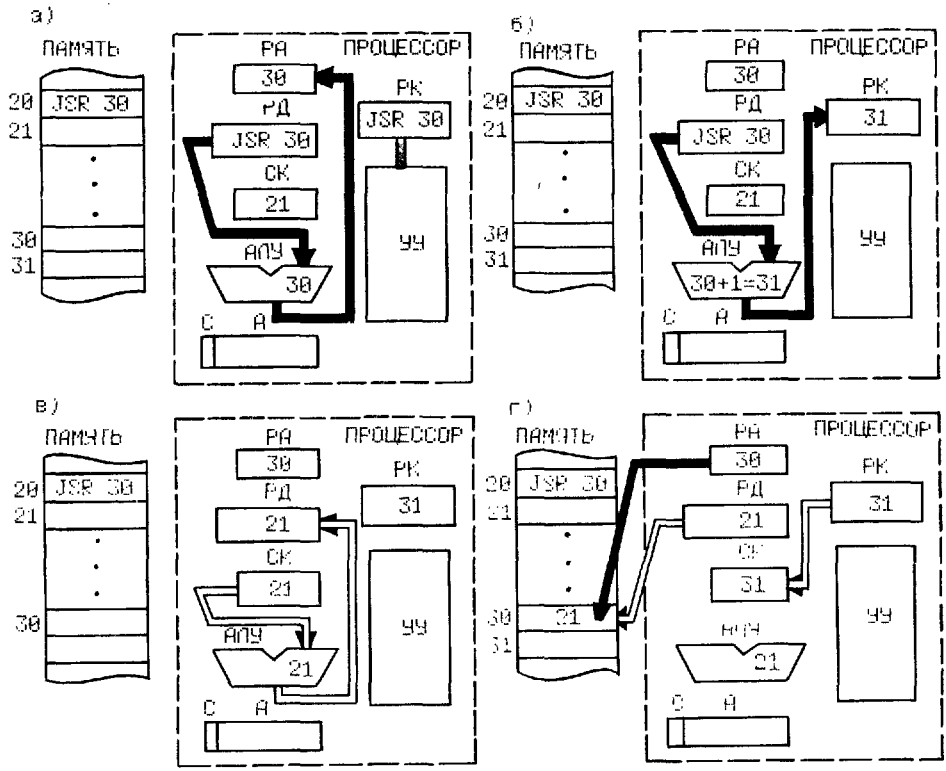


Рис. 2.10. Цикл «Исполнение» команды JSR 30

временное хранение в регистр команд (рис. 2.10, *б*). Потом содержимое счетчика команд (адрес команды, которая должна выполняться после возврата из подпрограммы) пересылается в регистр данных (рис. 2.10, *в*) и далее в ячейку памяти, указываемую регистром адреса (рис. 2.10, *г*). Наконец, в счетчик команд переписывается из регистра команд адрес первой команды подпрограммы (рис. 2.10, *г*).

Таким образом организуются переход к выполнению команд подпрограммы (начиная с команды в ячейке 31) и запоминание (перед первой ее командой) адреса возврата из подпрограммы. Когда в конце подпрограммы будет выполнена команда BR (30), то в счетчик команд попадет адрес 21, сохраняемый в ячейке 30, и машина продолжит выполнение команд, расположенных за командой JSR 30.

Как было указано в начале параграфа, время отладки машинных программ можно сократить, если знать, как реализуются команды ЭВМ, и иметь возможность выполнять отлаживаемую программу по отдельным командам, а иногда и по машинным циклам. Такая возможность предоставлена пользователю базовой ЭВМ. Он может выполнять программу не только по командам и машинным циклам, но и по отдельным тактам, т. е. проследить процесс выполнения каждой из команд, узнать, какие регистры использует процессор при их выполнении, какая информация размещается в этих регистрах и как реагирует микропрограммное устройство управления на изменение содержимого тех или иных регистров.

С процессом отладки можно ознакомиться в параграфе 3.1, где дано описание отладочного пульта базовой ЭВМ.

Глава 3

ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА ИНФОРМАЦИИ В БАЗОВОЙ ЭВМ

3.1. Отладочный пульт базовой ЭВМ

К ЭВМ можно подключать большое число разнообразных устройств ввода-вывода или, как часто говорят, внешних устройств (ВУ). Часть этих устройств описана в гл. 7, а в гл. 8 и последующих параграфах данной главы рассмотрена организация связей между процессором и ВУ. Здесь же речь пойдет о простейшем из таких устройств — отладочном пульте. Отладочный пульт базовой ЭВМ (рис. 3.1) позволяет произвести ввод программы и данных, подробно ознакомиться с процессом выполнения этой программы, рассмотреть взаимодействие процессора с несколькими внешними устройствами и т. п.

На пульт выведены индикаторы состояния основных регистров машины, 16 ячеек памяти (среди которых находится адресуемая в текущий момент ячейка), двух устройств управления (контроллеров) устройств ввода информации и одного контроллера устройства вывода информации, а также индикаторы состояний устройства управления базовой ЭВМ.

Индикатор каждого регистра или ячейки памяти представляет собой горизонтально расположенную линейку лампочек, число которых соответствует числу разрядов в регистре (ячейке памяти). Единичное состояние разряда отображается горящей лампочкой; если содержимое разряда равно нулю, соответствующая лампочка не горит. Для удобства считывания содержимого регистра (ячейки памяти) их разряды объединяются в группы по тетрадам (эквивалентам шестнадцатеричных цифр).

Под индикаторами расположена панель с тумблерами и кнопками, используемыми для ввода команд и данных, а также управления работой машины. Рассмотрим, как с их помощью можно производить ввод и проверку программы, обеспечить ее выполнение по машинным циклам, командам или в автоматическом режиме и т. п.

Ручные операции. Все ручные операции выполняются только тогда, когда тумблер РАБОТА/ОСТАНОВ находится в положении ОСТАНОВ.

Ввод программы и (или) данных. При этом осуществляются следующие шаги:

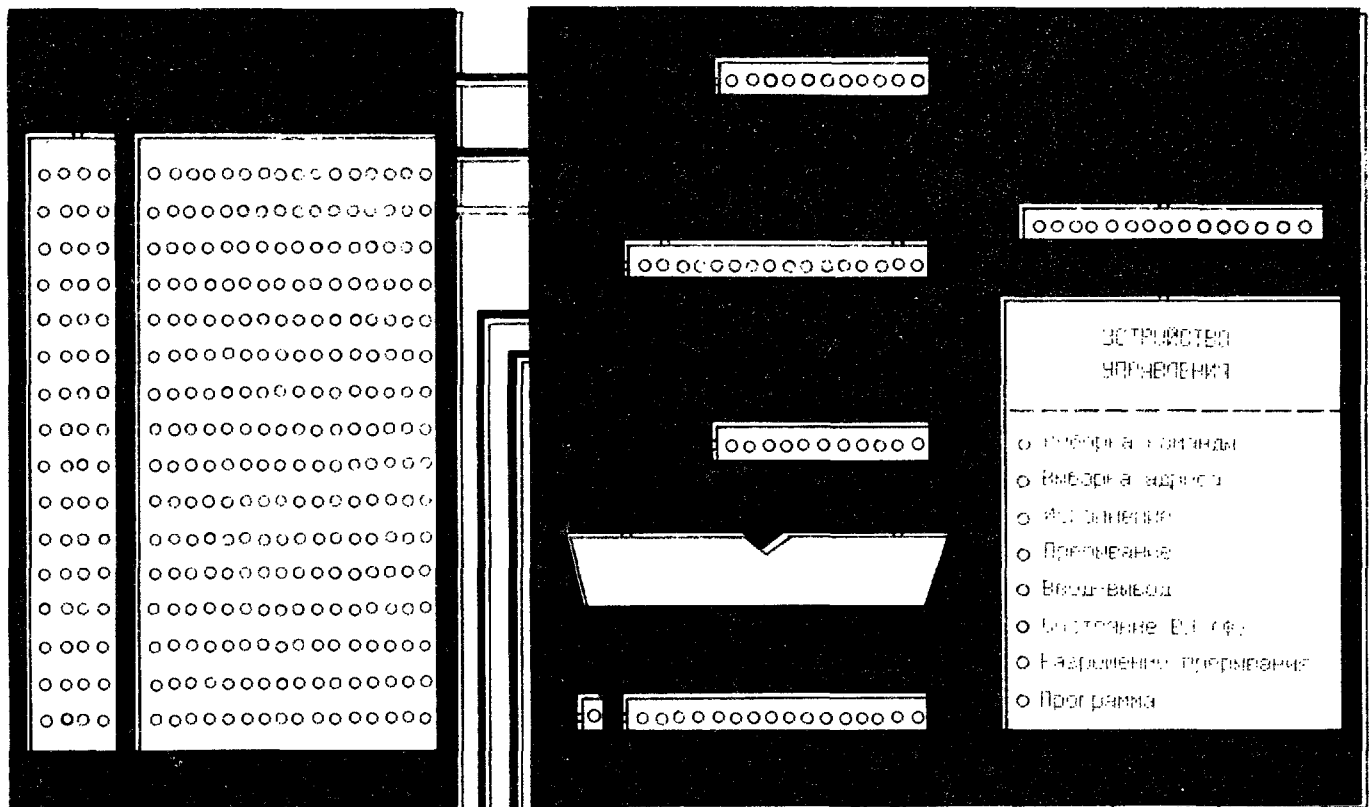
1) набрать на клавишном регистре (АДРЕС/ДАННЫЕ) адрес ячейки, в которую надо занести команду или число (для набора единицы тумблер должен быть установлен в положение от себя, а для набора нуля — в противоположное положение);

2) нажать кнопку ВВОД АДРЕСА (это приведет к переписи содержимого клавишного регистра в счетчик команд);

3) набрать на клавишном регистре код команды или числа;

4) нажать кнопку ЗАПИСЬ (это приведет к переписи содержимого счетчика команд в регистр адреса, а содержимого клавишного регистра — в регистр данных и далее в ячейку памяти, на которую указывает регистр адреса).

Так как в момент записи содержимое счетчика команд автоматически увеличивается на единицу, то для записи команд или данных в последовательные ячейки памяти требуется лишь выполнение пп. 3 и 4. Для записи же в разрозненные ячейки памяти каждый раз надо выполнять все четыре шага.


УСТРОЙСТВО
УПРАВЛЕНИЯ
Выборка задачи
Прерывание
Ввод-вывод
Программа

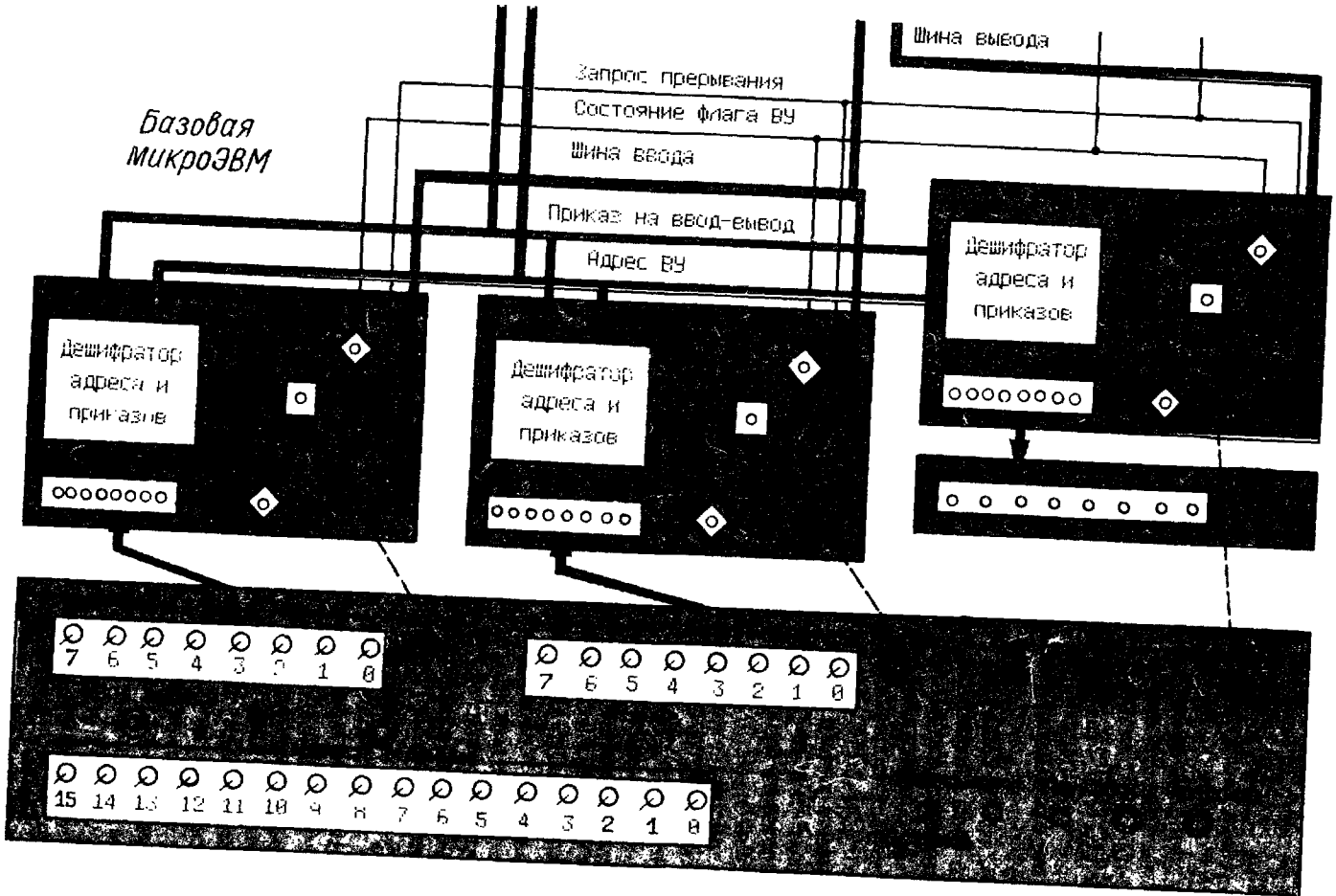


Рис. 3.1. Отладочный пульт базовой ЭВМ

Чтение программы и (или) данных. Два первых шага такие же, как при вводе команд (данных). Затем нажимается кнопка ЧТЕНИЕ, что приводит к переписи в регистр данных содержимого ячейки, на которую указывает регистр адреса, и к наращиванию на единицу содержимого счетчика команд. Таким образом, для просмотра содержимого следующей по порядку ячейки достаточно лишь снова нажать кнопку ЧТЕНИЕ.

Подготовка к исполнению программы. Перед подачей приказа на выполнение программы, введенной в память ЭВМ, в счетчик команд должен быть занесен адрес первой исполняемой команды этой программы (пусковой адрес). Для этого тумблер РАБОТА/ОСТАНОВ переводится в положение ОСТАНОВ, на клавишном регистре набирается пусковой адрес программы и нажимается кнопка ВВОД АДРЕСА.

Затем производится операция по установке требуемого режима выполнения программы: автоматического (команда за командой с остановом после выполнения команды HLT или по приказу с отладочного пульта) или пошагового (по отдельным командам, машинным циклам или тактам). Автоматический режим подготавливается переводом тумблера РАБОТА/ОСТАНОВ в положение РАБОТА. Для подготовки пошагового режима надо перевести этот тумблер в положение ОСТАНОВ и установить должным образом тумблеры ЦИКЛ и (или) ТАКТ. Выключение тумблеров (на себя) подготавливает режим работы по командам, включение тумблера ЦИКЛ — по машинным циклам, включение тумблера ТАКТ — по тактам, одновременное включение тумблеров ЦИКЛ и ТАКТ — по тактам.

Исполнение программы. Исполнение всей программы (подготовлен автоматический режим) можно инициировать двумя способами: нажатием кнопки ПУСК или кнопки ПРОДОЛЖЕНИЕ.

При нажатии кнопки ПУСК сначала очищаются (устанавливаются в нуль) аккумулятор, регистры адреса, данных, команд, переноса и флаги ВУ, устанавливаются состояние запрещения прерывания, режим ПРОГРАММА, начало цикла «Выборка команды» и начинается последовательное выполнение команд программы.

При нажатии кнопки ПРОДОЛЖЕНИЕ устанавливается режим ПРОГРАММА и начинается выполнение программы с той команды (того машинного цикла и его такта), на которой было прервано ее выполнение. Если до этого осуществлялись операции по занесению пускового адреса программы, то выполнение начнется с выборки команды, расположенной по этому адресу, при использовании «старых» (не равных нулю) содержимых основных регистров ЭВМ.

Так как во всех программах, рассмотренных в табл. 2.1, 2.7, 2.8 и примерах 2.4—2.6, пусковая команда (CLA) обнуляет аккумулятор, то выполнение этих программ можно с равным успехом инициировать как кнопкой ПУСК, так и кнопкой ПРОДОЛЖЕНИЕ.

Перед исполнением программы по командам (машинным циклам и/или тактам) может быть нажата кнопка ПУСК. При этом производится обнуление аккумулятора, регистров адреса, данных, команд, переноса и флагов ВУ, устанавливаются состояние запрещения прерывания и начало цикла «Выборка команды».

Последовательное выполнение отдельных команд или машинных циклов (в зависимости от подготовленного ранее режима) инициируется нажатием кнопки ПРОДОЛЖЕНИЕ.

Если установлен режим ТАКТ, то нажатие кнопки ПРОДОЛЖЕНИЕ приводит к выполнению лишь первого такта по переходу к выборке команды, после чего ЭВМ не реагирует на нажатие этой кнопки, но зато реагирует на нажатие кнопки ВВОД N. С помощью последней можно исполнить все

Таблица 3.1

Программа умножения двух чисел

Адрес	Содержимое			
	Мнемоническая форма	Шестнадцатеричная форма	Двоичная форма	
			Правильно	С ошибкой
0	CLA	F200	1111 0010 0000 0000	1111 0010 0000 0000
1	ADD 10	4010	0100 0000 0001 0000	0100 0000 0001 0000
2	ISZ 11	0011	0000 0000 0001 0001	0000 0000 0001 0001
3	BR 1	C001	1100 0000 0000 0001	1100 0000 0000 0000
4	HLT	F000	1111 0000 0000 0000	1111 0000 0000 0000
	...			
10		0078	0000 0000 0111 1000	0000 0000 0111 1000
11		FFFB	1111 1111 1111 1011	1111 1111 1111 1101

микрокоманды реализуемой команды, и когда будет исполнена последняя, ЭВМ перестанет реагировать на нажатие кнопки ВВОД N. Для выполнения по тактам следующей команды программы надо опять нажать один раз кнопку ПРОДОЛЖЕНИЕ, а затем нажимать кнопку ВВОД N до тех пор, пока ЭВМ не перестанет реагировать на эти приказы.

Из пошагового режима выполнения программы в любой момент можно перейти к ее автоматическому выполнению. Для этого тумблер РАБОТА/ОСТАНОВ переводится в положение РАБОТА и нажимается кнопка ПРОДОЛЖЕНИЕ.

Переход из автоматического режима в пошаговый осуществляется с помощью переключения тумблера РАБОТА/ОСТАНОВ в положение ОСТАНОВ. При этом ЭВМ заканчивает выполнение всех циклов текущей команды и останавливается. Теперь после просмотра на индикаторах пульта состояния основных устройств ЭВМ можно продолжить выполнение программы либо в автоматическом режиме (вернуть тумблер РАБОТА/ОСТАНОВ в положение РАБОТА и нажать кнопку ПРОДОЛЖЕНИЕ), либо по командам, машинным циклам или тактам.

Отладка программ. Выше (см. параграф 2.5) говорилось, что возможность пошагового выполнения программы позволяет ускорить процесс ее отладки. Для иллюстрации этого утверждения рассмотрим отладку программы в примере 2.5. Эта программа предназначена для умножения двух чисел (в примере сомножители равны 5 и $120=(78)_{16}$). В табл. 3.1 программа представлена в символической (как в примере 2.5), шестнадцатеричной и двоичной формах, причем последние две — правильная и ошибочная (вместо команды BR 1 записана команда BR 0 и вместо числа -5 — число -3).

После выполнения программы в автоматическом режиме аккумулятор (в котором должно располагаться произведение $120\times5=600=(258)_{16}$) содержал число $(0000\ 0000\ 0111\ 1000)_2=(78)_{16}=(120)_{10}$. Создалось впечатление, что в ячейке 11 вместо числа -5 записано число -1. Однако в процессе выполнения программы к содержимому этой ячейки последовательно добавлялись единицы, и в конце ее содержимое стало равно нулю. Следовательно, теперь уже не установить первоначального содержимого ячейки 11.

Повторное выполнение программы (после восстановления содержимого ячейки 11) дало тот же результат, и было принято решение выполнять программу по командам, занося результаты ее выполнения в табл. 3.2.

В процессе выполнения команд выясняется, что в ячейку 11 было занесено число -3 вместо числа -5. Однако это не могло дать результата, равного $(78)_{16}$, так как $(78)_{16}\times3\neq(78)_{16}$. Поэтому продолжаем поиск ошибки.

Таблица 3.2

Выполнение команд программы умножения двух чисел

Выполняемая команда		Содержимое регистров после выполнения команды						Комментарий
Адрес	Код	СК	РА	РК	РД	С	А	
000	F200	0001	0000	F200	F200	0	0000	0⇒А
001	4010	0002	0010	4010	0078	0	0078	$(78)_{16}$⇒А
002	0011	0003	0011	0011	FFFE	0	0078	В РД должно быть —4, а не —2, появившееся из-за того, что в ячейку 11 было занесено —3
003	C000	0000	0003	C000	C000	0	0078	Переход по адресу 0000
000	F200	0001	0000	F200	F200	0	0000	Аккумулятор очищен из-за ошибочного перехода на команду CLA

После окончания первого цикла выясняется, что вместо перехода на суммирование содержимых аккумулятора и ячейки 11 произведено обнуление аккумулятора. Теперь становится ясным, почему результат был равен $(78)_{16}$. Сколько бы циклов ни содержала программа, сумма, полученная в каждом из них (кроме последнего), будет затираться.

Могут возникнуть вопросы: «Надо ли было проверять программу путем последовательного выполнения команд? Не лучше ли просто внимательно проверить коды этих команд?» Если пользователь имеет правильный вариант кодировки, то тщательная проверка содержимого памяти помогает нахождению ошибок. Однако при большом объеме программы, когда приходится проверять сотни и тысячи 0 и 1, довольно быстро притупляется внимание и ошибки пропускаются. Кроме того, есть еще ошибки, которые возникли на этапе программирования: коды команд в памяти ЭВМ и на бумаге совпадают, но не соответствуют заданному алгоритму. Эта ситуация обнаруживается лишь при выполнении программы на ЭВМ.

3.2. Устройства ввода-вывода базовой ЭВМ

При разработке внешних устройств редко ориентируются на их подключение к определенному типу ЭВМ. Очень быстрое развитие вычислительной техники и, следовательно, частое обновление парка ЭВМ привели бы в этом случае к необходимости изменения тех схем ВУ, которые связывают его с конкретной ЭВМ, т. е. к изменению технологического процесса изготовления ВУ. Затраты на изготовление ВУ, пригодных для использования с разными типами ЭВМ (даже с теми, которые еще не разрабатывались), существенно уменьшаются, если при разработке ВУ ориентироваться только на лучшее выполнение ими заданных функций (печати, перфорации и т. п.). При этом для связи ВУ с ЭВМ можно использовать специальные управляющие устройства — контроллеры (англ. controller — управляющий, ревизор). Тогда для подключения ВУ к новому типу ЭВМ потребуется лишь разработать и изготовить новый контроллер.

В базовой ЭВМ, которая создавалась для обучения принципам функционирования ЭВМ, используются простейшие ВУ: два устройства ввода (ВУ-1 и ВУ-2) и одно устройство вывода (ВУ-3), размещенные на отладочном пульте (рис. 3.1). Устройство ввода представляет собой 8-разрядный тумблерный регистр и кнопку (ГОТОВ) для выдачи сигнала о том, что информация, набранная с помощью тумблеров, может быть считана в ЭВМ. Устройство вывода представлено 8-разрядным регистром, состояние которого высвечивается на отладочном пульте, и кнопкой (ВВОД N) для выдачи сигнала о том, что информация может быть принята в этот регистр.

Между ВУ и процессором установлены простейшие контроллеры, каждый из которых содержит регистр данных (для обмена данными между ВУ и процессором), дешифратор адреса (позволяющий выделить обращение к данному ВУ среди всех обращений к устройствам ввода-вывода, подключенным к процессору), дешифратор приказов (декодирующий приказ от процессора на выполнение тех или иных операций) и регистр состояния (в котором хранится информация о готовности ВУ к обмену данными с процессором). В контроллерах простейших ВУ обычно используются однобитовые регистры состояния, которые часто называют флагом или флажком. Это название распространяется и на контроллеры базовой ЭВМ.

Контроллеры связаны с процессором шинами ввода и вывода информации, шиной адреса и шиной управления, по которой передаются приказы от процессора в контроллеры ВУ и сведения о состоянии ВУ в процессор.

3.3. Программно-управляемая передача информации

Устройства ввода-вывода передают в ЭВМ и получают из нее, как правило, большой объем информации, который не может быть размещен только в регистрах процессора. Поэтому информация передается из ВУ в память ЭВМ и поступает на ВУ из ее памяти. При этом обмен может идти под управлением программы ЭВМ через регистры процессора (программно-управляемая передача данных) или под управлением специального внешнего устройства (контроллера прямого доступа в память), минуя процессор (передача данных при прямом доступе к памяти). Программно-управляемый обмен осуществляется малыми порциями, при прямом доступе к памяти информация передается большими блоками.

В базовой ЭВМ реализована только программно-управляемая передача данных.

1. Операция вывода предусматривает пересылку информации из аккумулятора процессора в регистр данных контроллера ВУ. После того как данные приняты этим регистром, начинают выполняться операции по их преобразованию в связанном с контроллером ВУ (например, печать соответствующего символа или изменение положения исполнительного органа). В базовой ЭВМ просто изменяется состояние индикатора регистра вывода ВУ-3.

2. Операция ввода предусматривает передачу данных из регистра данных контроллера ВУ в аккумулятор ЭВМ. В базовой ЭВМ устройство ввода имитируется с помощью набора тумблеров, состояние которых считывается в регистр данных при исполнении команды ввода.

При использовании программно-управляемого обмена должна быть составлена программа для пересылки данных из памяти ЭВМ в аккумулятор и далее в регистр данных контроллера ВУ (или из регистра данных контроллера ВУ в аккумулятор и затем в память ЭВМ).

В этой программе можно реализовать один из трех видов подобного обмена: синхронный, асинхронный и по прерыванию.

Синхронный обмен можно использовать лишь для связи с такими ВУ, для которых точно известно время выполнения операции (например, максимально возможное время, затрачиваемое на печать одного символа, пробивку одной строки перфоленты и т. п.). При этом программу обмена необходимо составлять так, чтобы команды на обмен шли с интервалами не меньшими, чем время выполнения одной операции обмена.

Это наиболее простой вид обмена, требующий минимальных затрат технических и программных средств. Однако при его использовании для передачи данных между ЭВМ и сравнительно медленными ВУ, как пра-

вило, не удается полностью загрузить ЭВМ на время между двумя пересылками данных. Так, например, электрическая пишущая машинка позволяет организовать вывод данных со скоростью не выше 7 символов в секунду, а за 1/7 секунды микроЭВМ может выполнить от 10 000 до 1 000 000 операций.

Асинхронный обмен широко используется в микроЭВМ. При его реализации интервал между командами передачи данных задается самим внешним устройством. Контроллеры этих устройств снабжаются регистром состояния, который информирует ЭВМ о готовности устройства к обмену информацией (например, электрическая пишущая машинка устанавливает сигнал готовности лишь тогда, когда она закончила печать предыдущего символа, в нее заправлены бумага и красящая лента). Программа обмена строится так.

1. Проверяется состояние устройства ввода-вывода.

2. Если устройство готово к обмену, то производится переход к п. 3. В противном случае вновь выполняется п. 1.

3. Производится передача очередной порции данных, выполняются действия по их обработке и (или) другие операции, и если обмен данными еще не закончен, осуществляется переход к п. 1.

Таким образом, при асинхронном обмене программист не должен знать о времени выполнения операции на ВУ. Последнее само информирует ЭВМ о завершении предыдущей операции и готовности к выполнению новой операции. Однако ЭВМ должна тратить время на ожидание момента готовности, а так как готовность проверяется программным путем (команда TSF B), то в это время ЭВМ не может выполнять никакой другой работы по преобразованию данных.

Обмен по прерыванию программы отличается от асинхронного обмена тем, что готовность ВУ к обмену проверяется при помощи аппаратных средств ЭВМ, а не программным путем. Это позволяет организовать работу ЭВМ следующим образом.

Машина исполняет какую-либо программу (назовем ее основной или фоновой), не связанную с обменом. Когда ВУ готово к приему или выдаче информации, оно посылает в ЭВМ сигнал готовности, воспринимаемый специальным блоком ЭВМ (например, контроллером прерываний). Этот блок приостанавливает (прерывает) исполнение основной программы и передает управление подпрограмме, организующей нужный вид обмена данными. Когда выполнение подпрограммы завершается, возобновляется работа ЭВМ по основной (временно прерванной) программе.

Различие между рассмотренными видами программно-управляемого обмена можно проиллюстрировать на следующем примере.

Пример 3.1. Вам поручено вскипятить воду в чайнике.

1. Безусловная операция (синхронный обмен). В 8 ч Вы поставили на плиту чайник с водой и, зная, что обычно он закипает через 10 мин, ровно в 8 ч 10 мин сняли его с плиты, не заботясь о том, закипела вода или нет.

2. Условная операция (асинхронный обмен). Вы поставили на плиту чайник с водой и примерно раз в минуту посматриваете за ним. Когда чайник закипит, Вы снимете его с плиты.

3. Операция по прерыванию. Вы поставили на плиту чайник с водой, надели на его носик свисток и стали читать книгу (фоновая работа). Когда засвистит свисток, вы снимете чайник с плиты и продолжите чтение книги.

В параграфах 3.4 и 3.5 рассмотрены наиболее употребительные виды обмена: асинхронный и по прерыванию программы.

Команды ввода-вывода базовой ЭВМ имеют одинаковый формат (см. рис. 2.2, *в*): 4-разрядное поле кода операции, 4-разрядное поле кода приказа и 8-разрядное поле кода выборки устройства ввода-вывода.

Код операции $(1110)_2$ обрабатывается в процессоре ЭВМ и служит для отличия команд ввода-вывода от других команд машины.

Код приказа используется для передачи контроллеру ВУ приказа на выполнение того или иного действия. Так как для этого кода отведены четыре разряда, то ЭВМ может переслать 16 различных команд внешнему устройству. Декодирование приказа осуществляется контроллером ВУ.

В базовой ЭВМ задаются три основных приказа: пересылка данных (IN B — ввод и OUT B — вывод), проверка готовности ВУ (TSF B) и сброс состояния готовности (CLF B), где B — адрес ВУ.

Код выборки устройства (адрес ВУ) используется для организации связи ЭВМ с одним из подключенных к ней внешних устройств. Этот код одновременно передается на все ВУ (их может быть $2^8=256$) и анализируется в их контроллерах. Когда последний распознает код, принадлежащий данному ВУ, он связывает ВУ с ЭВМ. Все другие устройства не будут связаны с ЭВМ и не реагируют на ее приказы.

В командах ввода-вывода (см. табл. 2.4) используется слово флаг. Так часто называют однобитовый регистр состояний ВУ, который устанавливается в единичное состояние, когда устройство готово к обмену информацией. Когда флажок ВУ сброшен (установка в нуль), ВУ занято: устройство вывода еще обрабатывает предыдущую команду, а устройство ввода не закончило подготовку данных.

Команда CLF B (E0xx, где xx — две шестнадцатеричные цифры кода выборки ВУ) служит для установки в исходное состояние ВУ с кодом выборки B. Команда CLF 02 сбрасывает (устанавливает в нуль) флажок в контроллере ВУ с кодом выборки 02.

По команде TSF B (E1xx) в РЕГИСТР СОСТОЯНИЙ ВУ (Ф в устройстве управления на рис. 3.1) помещается содержимое флажка ВУ с кодом выборки B. Затем процессор суммирует содержимое регистра Ф с содержимым счетчика команд. Следовательно:

при Ф=0 (устройство B не готово к обмену) выполняется следующая за TSF B команда программы;

при Ф=1 (устройство B готово к обмену) следующая за TSF B команда пропускается и выполняется команда, расположенная через одну за TSF B.

Команда IN B (E2xx) служит для пересылки в восемь младших разрядов аккумулятора содержимого регистра данных контроллера ВУ с кодом выборки B.

Команда OUT B (E3xx) служит для пересылки содержимого восьми младших разрядов аккумулятора в регистр данных устройства с кодом выборки B.

Когда в процессе работы ЭВМ устройство управления обнаруживает, что в регистре команд и одновременно в регистре данных находится команда с кодом операции $(1110)_2$, производится переход в режим ВВОД-ВЫВОД и на все контроллеры ВУ передается содержимое 12 младших разрядов регистра данных (см. рис. 3.1). Дешифраторы адреса всех контроллеров ВУ декодируют это содержимое, но срабатывает тот из них, код (адрес) которого совпадает с адресом, установленным в команде.

Этот дешифратор открывает вентили (см. параграф 1.2) для передачи информации на дешифратор приказов, который декодирует содержимое 8—11-го разрядов регистра данных и выдает приказ на выполнение одной из перечисленных выше команд: открывает вентиль для передачи в ЭВМ состояния флага (TSF), обнуляет флаг (CLF) или открывает вентили, связывающие аккумулятор с регистром данных контроллера (IN или OUT).

Для передачи в КОНТРОЛЛЕР ПРЕРЫВАНИЙ состояния флажков всех ВУ используется линия «Запрос прерывания», а для передачи в РЕ-

ГИСТР СОСТОЯНИЙ ВУ (Ф) состояния флажка опрашиваемого ВУ — линия «Состояние флага». Данные обычно передаются по одной шине данных, которая в базовой ЭВМ (для упрощения описания) представлена двумя шинами: шиной ввода и шиной вывода.

Программно-управляемый обмен осуществляется малыми порциями — обычно байтами. Такая 8-разрядная единица информации позволяет закодировать до 256 (2^8) различных символов, передаваемых между ЭВМ и ВУ: цифры, буквы латинского и русского алфавитов, знаки математического действия и т. п. (см. табл. Б.1 и Б.2 приложения Б). Поэтому шины ввода и вывода базовой ЭВМ связывают 8-разрядные регистры данных контроллеров ВУ с восмью младшими разрядами аккумулятора.

Так как слово базовой ЭВМ имеет 16 разрядов, то в одном таком слове можно разместить для хранения коды двух символов. В связи с этим программы организации обмена между памятью ЭВМ и различными ВУ целесообразно составлять следующим образом.

1. При вводе первый символ, который попал в младшие разряды аккумулятора, сдвигается в его старшие разряды. Затем в младшие разряды аккумулятора принимается следующий символ, и только тогда производится пересылка содержимого аккумулятора (кодов двух символов) в память.

2. При выводе в аккумулятор пересылается из памяти слово, содержащее два выводимых символа. Затем на ВУ выводится символ, код которого расположен в младших разрядах аккумулятора, и после сдвига — второй символ.

3.4. Асинхронный обмен

При асинхронной передаче данных необходимо сначала убедиться, что ВУ готово к обмену (включено и установлено в исходное состояние или уже закончило выполнение предыдущей операции), и лишь тогда давать команду на обмен (ввод или вывод).

О своей готовности к обмену ВУ сообщает с помощью установки в единичное состояние флага в контроллере ВУ. Для выяснения состояния этого флага следует выполнить команду «Опрос флага» (TSF), и тогда по линии «Состояние флага» в регистр состояний ВУ будет передано содержимое флага того устройства, адрес которого указан в команде опроса (см. рис. 3.1). В зависимости от этого содержимого (0 или 1) будет соответственно выполняться либо следующая команда программы, либо команда, расположенная за следующей (следующая команда будет пропущена).

Пример 3.2. С помощью ВУ-1 записать в ячейку 006 коды символов слова «Да».

Программа для выполнения этого задания приведена в табл. 3.3. При работе по такой программе ЭВМ будет ожидать нажатия кнопки ГОТОВ (имитирующей появления готовности ВУ к обмену информацией). Поэтому до первого нажатия этой кнопки необходимо набрать на тумблерах ВУ-1 код символа Д, затем нажать кнопку и приступить к набору кода символа А. В процессе этого набора ЭВМ занята сбросом флага готовности ВУ, пересылкой кода символа Д в младшие разряды аккумулятора и сдвигом этого кода в старшие разряды, чтобы подготовиться к приему кода следующего символа.

Так как быстродействие ЭВМ во много раз превышает быстродействие человека, то к моменту окончания набора кода символа А машина не только закончит действия по обработке символа Д, но и тысячи раз проверит содержимое флага ВУ-1. В связи с этим после набора кода можно сразу же нажимать кнопку ГОТОВ, т. е. давать команду на пересылку этого кода в аккумулятор ЭВМ. Теперь в аккумуляторе находится код слова «Да» и его содержимое можно пересылать в заданную ячейку.

Здесь следует напомнить, что состояние флагов подключенных к ЭВМ контроллеров ВУ пересылается в процессор по двум линиям: «Запрос прерывания» и «Состояние флага» (см. рис. 3.1). Линия «Запрос прерывания» непосредственно связана со всеми флагами контроллеров ВУ, а линия «Состояние флага» — через вентили, которые открываются лишь по коман-

Таблица 3.3

Подпрограмма ввода кодов двух символов

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
5	FFF8		Константа —8, используемая для сдвига
6	0000		Ячейка для записи слова «Да»
	...		
20	E101	TSF 1	Опрос флага контроллера ВУ-1 и повторение этой операции, если ВУ не готово к обмену (флаг=0)
21	C020	BR 20	
22	E204	IN 1	Это действие осуществляется лишь после нажатия кнопки ГОТОВ у ВУ-1, т. е. когда при выполнении TSF 1 выясняется, что флаг=1 и пропускается BR 20. По команде IN 1 содержимое регистра данных контроллера ВУ-1 пересылается в восемь младших разрядов аккумулятора
23	E001	CLF 1	Сброс готовности ВУ-1 для предотвращения считывания кода с ВУ-1 до тех пор, пока на нем не будет установлен код символа А и об этом не будет сообщено нажатием кнопки ГОТОВ
24	F600	ROL	Код первого введенного символа (Д) сдвигается на восемь разрядов влево, и освобождается место для ввода следующего символа
25	0005	ISZ 5	
26	C024	BR 24	
27	E101	TSF 1	Опрос флага контроллера ВУ-1 ... (см. комментарий к командам 20 и 21)
28	C027	BR 27	
29	E201	IN 1	Ввод кода символа, установленного на тумблерах ВУ-1 (если подан сигнал готовности ВУ-1)
2A	E001	CLF 1	Сброс готовности ВУ-1
2B	3006	MOV 6	Пересылка слова «Да» в ячейку 006
2C	F000	HLT	Останов ЭВМ

дам «Опрос флага» (TSF). При асинхронном обмене используется линия «Состояние флага». Поэтому нажатие любой другой кнопки ГОТОВ, кроме той, которая относится к устройству с адресом, указанным в команде TSF, не приведет к выполнению действий по вводу (или выводу) информации, хотя на линии «Запрос прерывания» и появится сигнал готовности.

3.5. Обмен по прерыванию программы

Передача данных с прерыванием программы особенно удобна при работе с низкоскоростными ВУ (пишущими машинками, перфораторами, телетайпами и т. п.), а также в ситуациях, когда момент передачи данных в ЭВМ непредсказуем, например при работе с каналами связи. Основной характерной чертой рассматриваемой передачи является то, что обмен данными между ЭВМ и ВУ инициируется сигналом с ВУ. Для реализации данного типа обмена необходимо заменить программный цикл ожидания готовности ВУ при асинхронной передаче (TSF, BR) аппаратной проверкой наличия внешнего прерывания, т. е. сигнала готовности по линии «Запрос прерывания» (см. рис. 3.1).

В этом случае ЭВМ может выполнять любую программу (будем называть ее основной), а когда с ВУ поступит запрос прерывания (запрос на передачу или прием данных), приостановить (прервать) выполнение этой программы на время осуществления обмена данными. Все действия по приостановке программы и переходу к подпрограмме обмена реализуются в базовой ЭВМ с помощью контроллера прерываний, входящего в состав устройства управления (см. рис. 3.1).

По командам «Разрешение прерывания» (E1) или «Запрещение прерывания» (D1) контроллер прерываний переходит в одно из двух состояний, в которых он соответственно реагирует или не реагирует на сигналы готов-

ности ВУ, передаваемые по линии «Запрос прерывания». Если контроллер установлен в состояние разрешения прерывания, то последовательность шагов по осуществлению рассматриваемой передачи данных обычно имеет такой вид.

Шаг 1. По завершении выполнения текущей команды основной программы управление передается контроллеру прерывания. Если в этот момент на линии «Запрос прерывания» нет сигнала о единичном состоянии флага какого-либо из ВУ, то начинаются выборка и исполнение следующей команды основной программы и данный шаг повторяется. При наличии запроса прерывания от ВУ выполняется второй шаг.

Шаг 2. Контроллер прерываний переходит в состояние запрещения прерывания, в ячейку памяти с адресом 0 заносится содержимое счетчика команд (адрес следующей команды основной программы, которая выполнялась бы в отсутствие прерывания), и управление передается команде, расположенной в ячейке 1.

Два последних действия имитируют выполнение команды «Обращение к подпрограмме» (JSR 0), которая как бы вставлена между только что выполненной и следующей командами основной программы (см. параграф 2.4). Такую подпрограмму, текст которой должен написать сам пользователь, обычно называют подпрограммой обработки или обслуживания прерываний. Она всегда начинается в ячейке с адресом 1 (в ячейке 0 хранится адрес возврата из подпрограммы) и может иметь самый различный характер в зависимости от набора ВУ, подключенных к ЭВМ, и принципов построения основной программы. Чаще всего, однако, это подпрограмма требует использования аккумулятора, и поэтому одним из первых действий является запоминание содержимого аккумулятора и регистра переноса. Затем выясняется источник прерывания (путем опроса флагов определяется ВУ, флаг которого установлен в единичное состояние) и выполняется передача данных. Заканчивается подпрограмма восстановлением содержимого аккумулятора и регистра переноса, разрешением прерывания и выходом из подпрограммы на команду, адрес которой хранится в ячейке 0. Эти функции определяют содержание следующих шагов.

Шаг 3. Производится запоминание в памяти содержимого аккумулятора и регистра переноса. Для этого требуются минимум три команды: пересылка содержимого аккумулятора в специально отведенную буферную ячейку (MOV B1), циклический сдвиг (обычно влево) для того, чтобы содержимое регистра переноса попало в аккумулятор (ROL), и запись этого содержимого в другую буферную ячейку (MOV B2). Таким образом, необходимый минимум информации о прерванной задаче оказывается обеспеченным - в ячейке 0 хранится адрес продолжения задачи, а в буферных ячейках запасены состояния двух других основных регистров машины в момент прерывания.

Шаг 4. Осуществляется поиск источника прерывания. Для этого в любой (наиболее целесообразной) последовательности опрашиваются флаги всех ВУ (команда TSF). При обнаружении ВУ с установленным флагом (флаг=1) выполняется переход к тому участку подпрограммы, в котором описаны действия по обмену информацией между процессором и этим внешним устройством.

Шаг 5. Выполняются передача данных и их предварительная обработка (если это необходимо).

Шаг 6. Восстанавливается содержимое регистра переноса и аккумулятора. Для этого требуются минимум пять команд: очистка аккумулятора (CLA), вызов содержимого буферной ячейки B2 в очищенный аккумулятор (ADD B2), циклический сдвиг вправо для восстановления содержимого

регистра переноса (ROR), очистка аккумулятора (CLA) и вызов содержимого буферной ячейки B1 в очищенный аккумулятор (ADD B1).

Шаг 7. Контроллер прерываний вновь переводится в состояние разрешения, и осуществляется возврат к выполнению прерванной (основной) программы. Для этого последовательно выполняются команды E1 («Разрешение прерывания») и BR (0) — переход к команде с адресом, хранимым в ячейке 0 (см. параграф 2.4). Здесь следует отметить, что команда E1 должна обязательно располагаться непосредственно перед командой выхода из подпрограммы — BR (0). Это делается на случай очередного требования прерывания во время обработки рассматриваемого. Если между командами E1 и BR (0) поместить хотя бы одну команду, то после ее выполнения может произойти переход к обработке прерывания, содержимое ячейки 0 заменится новым адресом возврата, и таким образом путь возвращения к прерванной (основной) программе будет разрушен.

В заключение напомним следующее.

1. С помощью команды D1 контроллер прерываний устанавливается в состояние запрещения прерывания, и ЭВМ не реагирует на запросы обмена от любых внешних устройств. В то же состояние переходит этот контроллер и после нажатия кнопки ПУСК, поэтому для обеспечения обмена по прерыванию программы в последней обязательно должна присутствовать команда E1 («Разрешение прерывания»).

2. Прерывание программы наступает при наличии следующих условий: контроллер прерываний находится в состоянии разрешения прерывания, появился запрос на прерывание (флаг какого-либо ВУ установился в единичное состояние) и окончилось выполнение любой текущей команды, кроме команд D1 («Запрещение прерывания») и E1. Если в момент запроса на прерывание исполнялась команда E1, то состояние прерывания наступит после выполнения следующей за E1 командой (см. шаг 7).

3. В зависимости от конкретных условий подпрограмма обработки прерываний может иметь любой вид, но она всегда должна начинаться с команды, расположенной в ячейке с адресом 1, и заканчиваться командами E1 и BR (0), следующими друг за другом. Команда E1 служит для восстановления состояния разрешения прерывания, которое было автоматически заменено на запрещение прерывания при входе в подпрограмму, для того чтобы во время обработки текущего прерывания ЭВМ не реагировала на последующие.

Пример 3.3. Составить программу, которая периодически (с периодом в три цикла команды) наращивает на единицу содержимое аккумулятора. Восемь младших разрядов последнего должны выводиться на ВУ-3 по запросу последнего (нажатие кнопки ВВОД N), а по запросу с ВУ-1 код, набранный на его тумблерах, должен помещаться в ячейку с адресом 25.

Таблица 3.4

Основная программа решения задачи примера 3.3

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
20 21	FA00 F200	E1 CLA	Установка состояния разрешения прерывания Очистка аккумулятора
22 23 24	F800 F100 C022	INC NOP BR 22	Цикл для наращивания содержимого аккумулятора
25	0000		Ячейка для хранения кодов, поступающих из ВУ-1

Основная программа решения такой задачи достаточно проста. Она может начаться с установки разрешения прерывания (EI) и очистки аккумулятора (CLA). Затем должен быть организован цикл для наращивания на единицу содержимого аккумулятора. Этот цикл может содержать всего две команды: «Инкремент аккумулятора» (INC) и команду безусловного перехода (BR) для возврата к INC. Однако по заданию такой цикл должен состоять из трех команд, и поэтому между INC и BR можно поставить команду NOP («Нет операции»). Подобная программа приведена в табл. 3.4.

Если команды этой программы занести в память ЭВМ, установить в счетчик команд пусковой адрес 0020 и нажать кнопку ПУСК, то начнет выполняться бесконечный цикл наращивания содержимого аккумулятора. В этом цикле все время будут повторяться команды: INC, NOP, BR 22, INC, NOP, BR 22, INC и т. д. Когда же на пульте управления (см. рис. 3.1) будет нажата любая из трех кнопок (ГОТОВ, ГОТОВ или ВВОД N), то произойдет передача управления команде, расположенной в ячейке 1, а в ячейку 0 будет записан адрес той из вышеперечисленных команд, которая выполнялась бы при отсутствии запроса на прерывание. Так осуществляется переход к подпрограмме обработке прерываний, которая должна быть расположена начиная с ячейки 1.

Таблица 3.5

Первый вариант подпрограммы обработки прерываний для примера 3.3

Адрес	Содержимое		Комментарий	
	Код	Мнемоника		
0	0000		Ячейка для хранения адреса возврата (этот адрес будет занесен сюда на 2-м шаге)	
1	C030	BR 30	Первая команда подпрограммы — переход к основному ее тексту, расположенному в ячейках 30—4C	
	...			
30	304B	MOV 4B	Сохранение в буферных ячейках 4B и 4C содержимого аккумулятора и регистра переноса	Шаг 3
31	F600	ROL		
32	304C	MOV 4C		
33	E101	TSF 1	Опрос флага ВУ-1. Если он сброшен, то переход к опросу флага ВУ-3. В противном случае переход на ввод данных из ВУ-1	Шаг 4
34	C036	BR 36		
35	C039	BR 39		
36	E103	TSF 3	Опрос флага ВУ-3. Если он сброшен, то переход к сбросу флага ВУ-2. В противном случае переход на вывод данных в ВУ-3	
37	C043	BR 43		
38	C03E	BR 3E		
39	F200	CLA	Ввод данных из ВУ-1, пересылка их в ячейку 25, сброс флага ВУ-1, переход к восстановлению содержимого основных регистров и выходу из подпрограммы	Шаг 5
3A	E201	IN 1		
3B	E001	CLF 1		
3C	3025	MOV 25		
3D	C044	BR 44		
3E	F200	CLA	Пересылка в аккумулятор содержимого буферной ячейки 4B, вывод на ВУ-3 восьми младших разрядов аккумулятора, сброс флага ВУ-3, переход к восстановлению регистров и выходу	
3F	404B	ADD 4B		
40	E303	OUT 3		
41	E003	CLF 3		
42	C044	BR 44		
43	E002	CLF 2	Очистка флага ВУ-2	
44	F200	CLA	Восстановление содержимого регистра переноса и аккумулятора	Шаг 6
45	404C	ADD 4C		
46	F700	ROR		
47	F200	CLA		
48	404B	ADD 4B		
49	FA00	EI	Возобновление состояния разрешения прерывания и выход из подпрограммы	Шаг 7
4A	C800	BR (0)		
4B	0000		Ячейки для сохранения содержимого аккумулятора и регистра переноса	
4C	0000			

В табл. 3.5 приведен один из возможных вариантов такой подпрограммы, построенной по стандартной схеме: запоминание содержимого основных регистров (шаг 3), отыскание источника прерывания (шаг 4), передача данных (шаг 5), восстановление содержимого основных регистров (шаг 6) и, наконец, организация выхода из подпрограммы (шаг 7). Подпрограмма предусматривает реакцию не только на запросы от ВУ-1 и ВУ-3, но и на запрос от ВУ-2, который может возникнуть от непреднамеренного нажатия на кнопку ГОТОВ около этого «устройства». В последнем случае производится очистка флага ВУ-2, что эквивалентно отмене непредусмотренного запроса.

Так как индексные ячейки 008–00F часто используются в циклических программах, то их нецелесообразно занимать под команды программы обработки прерываний. Поэтому в примере 3.3 основной текст такой подпрограммы размещен начиная с ячейки 30, а переход к ней осуществляется по команде BR 30, размещенной в ячейке 001.

Количество команд подпрограммы обработки прерываний может быть существенно сокращено, если учесть особенности решаемой задачи.

1. Накопленное в основной программе содержимое аккумулятора может быть испорчено (заменено на другое значение) лишь во время выполнения операции ввода данных с ВУ-1.

2. При любых (необходимых для решения задачи) взаимодействиях с ВУ-1, ВУ-2 или ВУ-3 содержимое регистра переноса не затрагивается.

Таблица 3.6

Второй вариант подпрограммы обработки прерываний для примера 3.3

Адрес	Содержимое		Комментарий
	Код	Мнемоника	
0	0000		Адрес возврата из подпрограммы
1	C030	BR 30	Переход к выполнению подпрограммы
	...		
30	E101	TSF 1	Опрос флага ВУ-1 и переход к опросу флага ВУ-3, если флаг ВУ-1 сброшен
31	C03C	BR 3C	
32	303B	MOV 3B	Сохранение содержимого аккумулятора в ячейке 3B
33	F200	CLA	Ввод данных из ВУ-1 и пересылка их по заданному адресу (ячейка 25)
34	E201	IN 1	
35	E001	CLF 1	
36	3025	MOV 25	
37	F200	CLA	Восстановление содержимого аккумулятора
38	403B	ADD 3B	
39	FA00	EI	Возобновление состояния разрешения прерывания и выход из подпрограммы
3A	C800	BR (0)	
3B	0000		Буфер для сохранения содержимого аккумулятора
3C	E103	TSF 3	Опрос флага ВУ-3 и переход к сбросу флага ВУ-2, если флаг ВУ-3 сброшен
3D	C041	BR 41	
3E	E303	OUT 3	Вывод на ВУ-3 восьми младших разрядов аккумулятора и переход к завершающим командам подпрограммы
3F	E003	CLF 3	
40	C039	BR 39	
41	E002	CLF 2	Сброс флага ВУ-2 и переход к завершающим командам подпрограммы
42	C039	BR 39	

3. На ВУ-3 должно выводится накопленное в основной программе содержимое аккумулятора, и оно не может быть испорчено во время этого вывода.

Вариант подпрограммы обработки прерываний, составленный с учетом перечисленных особенностей, приведен в табл. 3.6. Он короче первого на 10 команд (более чем 30 %-ная экономия).

Глава 4

МИКРОПРОГРАММНОЕ УСТРОЙСТВО УПРАВЛЕНИЯ БАЗОВОЙ ЭВМ

4.1. Многоуровневые ЭВМ и их микропрограммный уровень

В настоящее время большинство пользователей ЭВМ начинает общение с этой машиной на одном из алгоритмических языков (ФОРТРАНе, АЛГОЛе, БЕЙСИКе и т. п.), считая, что такой язык является языком машины. Вполне вероятно, что скоро будут конструировать подобные машины, но сейчас аппаратная реализация ЭВМ с алгоритмическими машинными языками весьма сложна и дорога. Поэтому машинные языки почти всех современных ЭВМ достаточно примитивны (похожи на язык базовой ЭВМ), что делает непосредственное использование таких языков неудобным и затруднительным. Возможность же исполнения на ЭВМ программы, написанной на алгоритмическом языке, обеспечивается с помощью специальных системных программ (компиляторов или интерпретаторов), осуществляющих перевод пользовательских программ на машинный язык ЭВМ.

Существуют два способа организации процесса перевода исходной программы с алгоритмического на машинный язык и процесса ее выполнения.

Первый способ, получивший название компиляции, заключается в том, что процесс выполнения алгоритма осуществляется лишь после завершения процесса перевода исходной программы. При этом в процессе выполнения полученной в результате перевода машинной программы уже не нужна не только исходная программа, но и программа-компилятор, осуществившая ее перевод в машинное представление.

Второй способ, получивший название интерпретации, заключается в том, что каждый оператор исходной программы заменяется программой-интерпретатором на эквивалентную последовательность машинных команд. Эта последовательность тут же выполняется, после чего переводится следующий оператор исходной программы, реализуются полученные при его переводе команды и т. д. Следовательно, во время решения задачи машине нужны и исходная программа, и программа-интерпретатор.

Затраты на создание компилятора (интерпретатора) и время на процесс перевода программы в значительной мере определяются сходством компилируемого и получаемого языков. Поэтому распространенные алгоритмические языки и более сложные языки программных систем не сразу переводят на язык машинных команд. Существует определенная иерархия языков программирования, в которой каждый более сложный язык базируется на предшествующем.

Мы уже сталкивались с одним из простейших языков, который очень близок к командам ЭВМ и обычно служит промежуточным языком между ними и алгоритмическими языками. Это язык символического кодирования

команд, часто называемый языком ассемблера. Языки ассемблеров (разработанные для каждого типа ЭВМ) – это первые средства автоматизации программирования в вычислительной технике. В них допускается использование символических имен и меток (адресов). Компиляторы с таких языков называются ассемблерами. Они отводят определенные ячейки памяти для символических переменных, организуют связи между различными частями программы, что резко облегчает программирование по сравнению с программированием на уровне команд.

Возможность реализации на любой ЭВМ программ, написанных на языках, отличающихся от ее машинного языка, лишний раз подчеркивает универсальность ЭВМ. Сейчас любой квалифицированный пользователь ЭВМ может создать свой собственный язык для описания каких-либо специфических задач (моделирования ЭВМ, управления прибором или ходом эксперимента, обучения игре в шахматы и т. п.) и написать для этого языка, например, интерпретатор на языке ФОРТРАН или БЕЙСИК.

Человек, работающий с ЭВМ на том или другом языке (или обучающийся игре в шахматы), может не знать, выполняется ли его программа шаг за шагом интерпретатором, который, в свою очередь, выполняется другим интерпретатором, или же она осуществляется непосредственно аппаратными средствами (его обычно интересует лишь результат выполнения программы). Чаще всего ему кажется, что язык, на котором он общается с ЭВМ,– это ее машинный язык. Следовательно, разным пользователям одной и той же ЭВМ может казаться, что они работают на разных вычислительных машинах. Отсюда появились понятия: виртуальная (кажущаяся) ЭВМ и многоуровневая ЭВМ.

Многоуровневая ЭВМ – это вычислительная машина, имеющая средства для работы с n различными уровнями языков программирования (рис. 4.1). Нижний язык, или уровень, является наиболее простым, верхний – наиболее сложным. Такую машину можно рассматривать как n различных виртуальных машин, каждая из которых имеет свой машинный язык. Сложность аппаратурной реализации этих виртуальных машин возрастает по мере усложнения языка (увеличения номера уровня).

Если рассматривать с этих позиций базовую ЭВМ, то окажется, что язык команд, описанных в гл. 2, не является языком самого нижнего уровня. На ее нижнем уровне выполняются элементарные действия (микрооперации) над словами информации. Ряд из этих действий уже рассматривался при

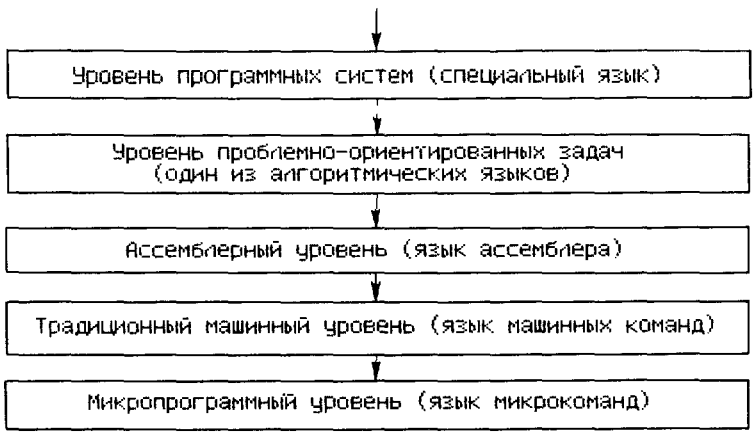


Рис 4.1. Многоуровневая ЭВМ

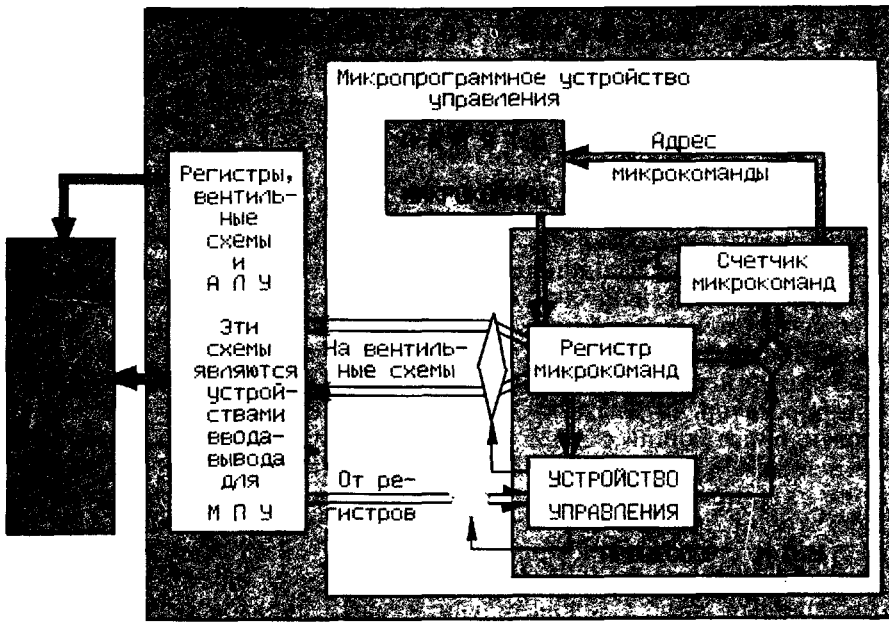


Рис. 4.2. Структура микропрограммного устройства управления и его связь с другими устройствами базовой ЭВМ

описании выполнения команд в параграфах 2.1 и 2.5 (пересылка содержимого одного регистра в другой регистр, проверка бита в каком-либо регистре и т. п.).

Управление порядком следования микроопераций осуществляется с помощью устройства управления базовой ЭВМ, которое, в свою очередь, является очень простой ЭВМ. Для этой ЭВМ регистры и вентильные схемы базовой ЭВМ служат как бы устройствами ввода и вывода (рис. 4.2). Программа работы такой ЭВМ — микропрограммное устройство управления (МПУ) — называется микропрограммой, а ее команды, содержащие информацию об элементарных действиях, подлежащих выполнению в течение одного рабочего такта ЭВМ,— микрокомандами.

Микропрограмма хранится в постоянном запоминающем устройстве — памяти микрокоманд (МП). В каждом такте работы ЭВМ из МП в регистр микрокоманд пересылается очередная микрокоманда, т. е. микрокоманда, на которую указывает счетчик микрокоманд (он же регистр адреса микрокоманд). Затем счетчик микрокоманд наращивается на единицу.

Если из памяти выбрана так называемая операционная микрокоманда, биты которой определяют нужный набор микроопераций, то состояние этих битов передается на вентильные схемы процессора. Производятся соответствующая настройка АЛУ (на суммирование, логическое умножение, инвертирование и т. п.) и пересылка через него содержимого одних регистров в другие регистры (см. рис. 2.2, 2.8, 2.9 или 2.10).

При выборке управляющей микрокоманды в устройство управления МПУ пересылается содержимое указанного в микрокоманде регистра ЭВМ, из него выделяется указанный в микрокоманде бит и сравнивается с определенным битом микрокоманды (с 0 или 1). Если результат сравнения положителен, то в счетчик микрокоманд пересылается из микрокоманды адрес, по которому должна выбираться следующая микрокоманда микропрограммы. В противном случае никаких пересылок не производится и в следующем

такте будет выполняться микрокоманда, расположенная вслед за исполняемой.

Следует ясно представлять, что у описываемой здесь микропрограммируемой ЭВМ имеются две памяти, два набора команд и две программы: традиционного машинного уровня (команды табл. 2.4) и микропрограммного уровня. Однако аппаратно реализованный процессор — только один, а архитектура машины формирует микропрограммный уровень.

Традиционный машинный уровень — это один из тех машинных уровней, описание которого можно найти в руководстве по эксплуатации ЭВМ. В рассматриваемом случае базовая ЭВМ рекламировалась бы как ЭВМ, обладающая памятью емкостью 2048 16-битовых слов и системой команд, состоящей из 28 команд, среди которых есть логическое умножение, сложение, вычитание, условные и безусловный переходы, обращение к подпрограмме и т. п. Она имеет два вида адресации (прямую и косвенную), к ней можно подключать до 256 внешних устройств и т. д. В руководстве можно найти упоминание о том, что машина микропрограммируема, однако описания микропрограммного уровня, памяти микрокоманд и самих микрокоманд там обычно не приводится.

Пользователю, желающему программировать на традиционном машинном уровне (и тем более на языках высокого уровня), в конечном счете безразлично, реализован ли традиционный уровень путем создания специальных электрических схем или же для этих целей используется микропрограмма, выполняемая на более низком машинном уровне. Однако в связи с появлением ряда микропроцессорных наборов, имеющих единственный язык — язык микрокоманд, возникла необходимость в ознакомлении пользователей с программированием на уровне микрокоманд.

4.2. Компоненты процессора и основные операции

В параграфах 1.2 и 2.1 уже давалось описание большинства компонентов процессора: регистров, шин, вентильных схем и генератора тактовых импульсов. Здесь подробнее рассмотрим способы передачи данных между регистрами, работу АЛУ и ряда вспомогательных схем.

Передача данных между регистрами. Это основная операция микропрограммного уровня. Можно указать четыре типа такой передачи данных: прямая, асимметричная, по частям и групповая (рис. 4.3). В случае прямой передачи содержимое N-битового регистра копируется в N-битовом регистре так, что i-й бит копии соответствует i-му биту оригинала. При асимметричной

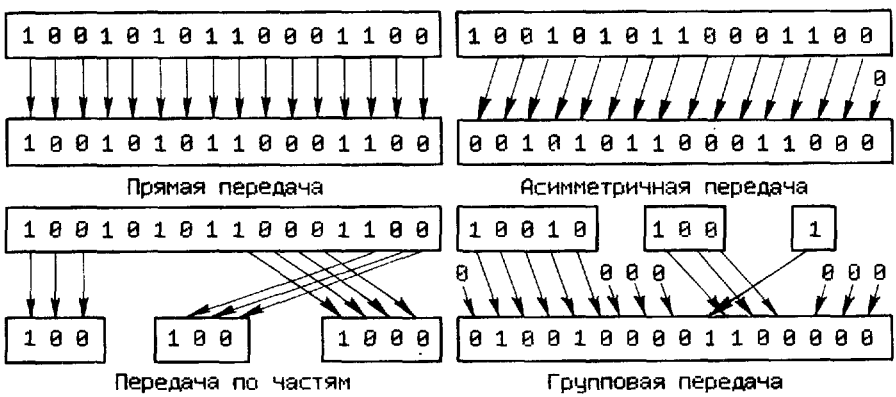


Рис. 4.3. Передача данных между регистрами

передаче i-й бит копии соответствует $(i+K)$-му биту оригинала, где K — положительное или отрицательное целое число. В этом случае для одного или нескольких разрядов регистра-приемника может не оказаться соответствующих битов из регистра-источника. Такие разряды заполняются нулями.

При передаче по частям содержимое одного регистра копируется (оказывается «разбросанным») в нескольких регистрах, причем порядок расположения битов может быть изменен. В случае групповой передачи в различные части регистра-приемника записываются копии содержимого нескольких регистров, порядок расположения битов также может быть изменен. Разряды, не имеющие соответствующих источников информации, заполняются нулями.

Передача данных между регистрами изменяет содержимое регистра-приемника, но сохраняет неизменным содержимое регистра источника. Для передачи используются шины различной ширины с встроенными в них вентильными схемами (рис. 1.3).

На рис. 4.4 показана работа устройства сдвига. Число, подлежащее циклическому сдвигу на 1 бит влево, помещается в аккумулятор. Затем выполняется асимметричная передача во внутренние регистры АЛУ: буферный регистр и однобитовый регистр *Б*. При этом младший бит буферного регистра заполняется не нулем, а содержимым регистра переноса *С*. После этого производится прямая передача из *Б* и буферного регистра в *С* и аккумулятор.

Следует заметить, что обе эти передачи осуществлялись с помощью различных шин. Одна из них (с асимметричной связью разрядов регистра-источника с регистром-приемником) служит для передачи информации из аккумулятора в буферный регистр, а другая — из буферного регистра (БР) в аккумулятор. Для того чтобы данная пара регистров могла осуществлять и сдвиг вправо, необходимо использовать еще один информационный канал (шину с вентильной схемой) с правой асимметрией.

Работа арифметико-логического устройства. АЛУ использует содержимое аккумулятора и регистра данных в качестве операндов для получения результата, который помещается в аккумулятор. Все арифметические и логические команды базовой ЭВМ (см. табл. 2.4) и вспомогательные арифметические операции (например, увеличение на единицу содержимого счетчика команд) можно выполнить с помощью АЛУ, структурная схема которого приведена на рис. 4.5. Кратко рассмотрим схемную реализацию отдельных микроопераций, выполняемых АЛУ, аккумулятором и регистром переноса (С) по управляющим сигналам У7—У17 и У22, поступающим на вентильные схемы *В7—В17* и *В22* (на рис. 4.5 управляющие сигналы У7—У17 и У22 не показаны).

Обратный код входных сигналов АЛУ может быть получен по схеме рис. 4.6, *а*), в которой изменение всех 0 на 1 и всех 1 на 0 осуществляется с помощью инверторов *СхНЕ*. При отсутствии управляющего сигнала (см. рис. 1.3) единица не проходит на выход *В7*, вентиль *ВИ2* закрыт, на выходе инвертора *НЕ* единичный сигнал и вентиль *ВИ1* открыт. В этом случае в узле *Вых.*— прямой код операнда, т. е. код, совпадающий с кодом в узле *Вх*. Когда на *В7* подается единичный управляющий сигнал, через него проходит *1*, создаваемая соответствующим постоянным напряжением (см. рис. 1.2), закрывается *ВИ1*, открывается *ВИ2* и в узел *Вых.* подается код, обратный по отношению к коду в узле *Вх*.

Сложение осуществляется с помощью типовой схемы сумматора (рис. 4.6, *б*). Функция, реализуемая одним из разрядов этого сумматора (разрядом с номером 1), представлена в виде табл. 4.1. Входными пере-

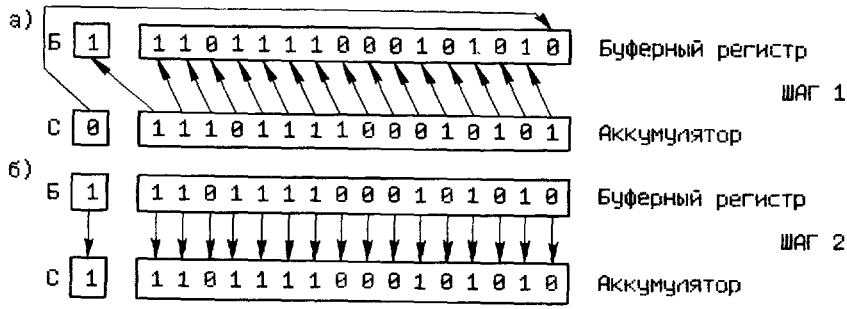


Рис. 4.4. Работа устройства сдвига на 1 бит влево

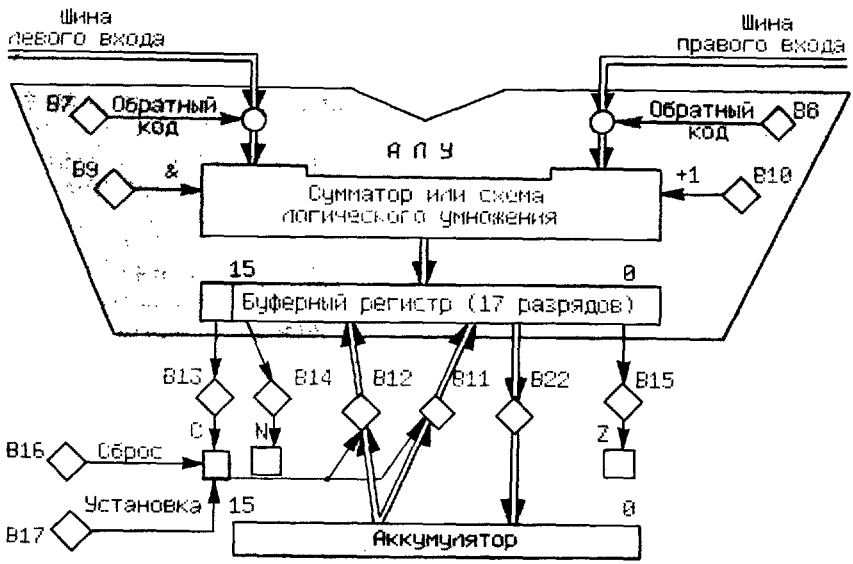


Рис. 4.5. АЛУ, аккумулятор и регистр переноса

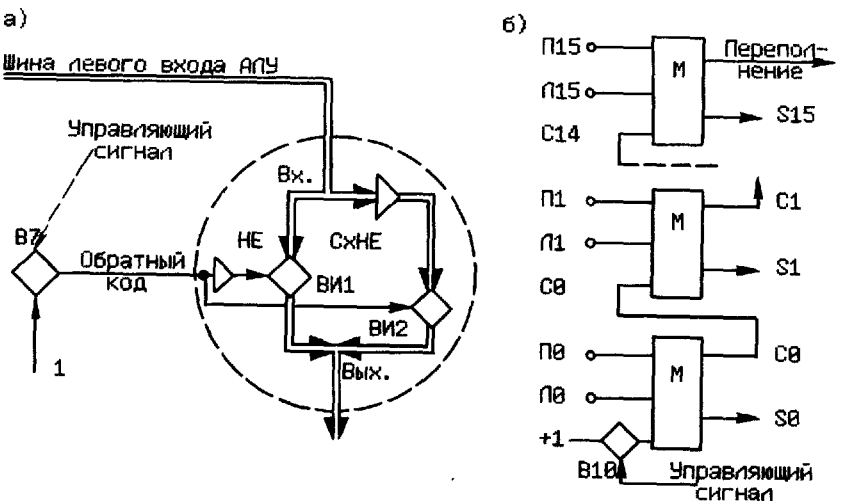


Рис. 4.6. Схемы получения обратного кода (*а*) и суммы (*б*)

Таблица 4.1

Таблица выходов для одного разряда двоичного сумматора SM

Перенос из предыдущего разряда *C0*	Слагаемые		Сумма *S1*	Перенос в следующий разряд *C1*
	Левый вход *Л1*	Правый вход *П1*		
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

менными таблицы являются *C0*, *Л1*, и *П1*, выходными переменными — *S1* и *C1*. При закрытом вентиле *B10* сумматор реализует зависимость S=Л+П.

С помощью управляющего сигнала, подаваемого на вентиль *B10*, можно обеспечить прибавление к сумме единицы: S=Л+П+1.

Перенос из старшего разряда (разряда с номером 15) сигнализирует о переполнении сумматора. Этот перенос записывается в старший разряд (разряд с номером 16) буферного регистра (см. рис. 4.5) и может быть переписан в регистр переноса С путем подачи управляющего сигнала на вентиль В13.

Следует отметить, что сумматор, вентильная схема и инвертор - это так называемые комбинационные схемы, т. е. схемы, в которых значения выходных переменных полностью определяются значениями входных переменных в данный момент времени. Следовательно, изменение любого из входных сигналов АЛУ сразу же приведет к изменению суммы. Это позволяет выполнять с помощью сумматора и схем получения обратного кода несколько различных микроопераций при различных сочетаниях управляющих сигналов, подаваемых на вентильные схемы *B1*—*B8* и *B10* (рис. 4.5 и 4.7):

1) суммирование содержимых двух регистров, один из которых подключен к левому входу АЛУ, а другой — к правому.

Для этого надо подать единичный управляющий сигнал на *B1* или *B2*, или *B3* и такой же сигнал на *B4* или *B5*, или *B6* (при выполнении команды ADD управляющие сигналы подаются на *B1* и *B4*);

2) суммирование содержимых двух регистров и 1 (три слагаемых).

Эта микрооперация отличается от предыдущей лишь тем, что одновременно с подачей управляющих сигналов на вентили, устанавливающие связь между сумматором и регистрами (где хранятся коды слагаемых), такой же сигнал должен подаваться на вентиль *B10*;

3) вычитание.

Например, при вычитании содержимого регистра данных из содержимого аккумулятора (команда SUB в табл. 2.4) надо одновременно подать единичные управляющие сигналы на *B1*, *B4*, *B8* и *B10*. Тогда к уменьшаемому прибавится обратный код вычитаемого и к этой сумме добавится единица, что эквивалентно суммированию уменьшаемого с дополнительным кодом вычитаемого (см. параграф 2.3);

4) добавление единицы к содержимому какого-либо регистра.

Эта микрооперация нужна для наращивания содержимого счетчика команд, выполнения команды INC (см. табл. 2.4) и в ряде других случаев. Для ее реализации необходимо подать управляющие сигналы на *B10* и вентильную схему, устанавливающую связь между сумматором и соответствующим регистром. При этом другой вход сумматора будет соединен с шиной, все вентильные схемы которой закрыты, что эквивалентно пересылке по шине кода числа 0;

5) вычитание единицы из содержимого какого-либо регистра.

Эта микрооперация нужна, например, для реализации команды DEC (см. табл. 2.4). Управляющие сигналы подаются на вентильные схемы *B4* и *B8*, т. е. содержимое аккумулятора суммируется с обратным кодом числа 0 или (что то же самое) с дополнительным кодом числа —1 (см. параграф 2.3);

6) инвертирование содержимого какого-либо регистра.

Такое преобразование уже использовалось в микрооперациях пп. 3 и 5. Для выполнения же, например, команды CMA (см. табл. 2.4) следует подать управляющие сигналы на *B4* и *B7*;

7) очистка какого-либо регистра (установка в нем кода числа 0).

Для этого надо выполнить две микрооперации: засылку нуля в БР (в данном такте закрыты все вентили с *B1* по *B10*) и пересылку содержимого БР в нужный регистр (открыт соответствующий вентиль на выходной шине БР, например *B22* при выполнении команды CLA).

Список подобных микроопераций можно было бы продолжить (например, занесение в регистр кода числа —1, получение сигнала переноса из старшего разряда сумматора при сложении двух чисел, равных —1, и т. д.), однако и приведенного перечня достаточно для подтверждения следующего положения:

чтобы интерпретируемая машина выполняла программу, необходимо в определенной последовательности открывать и закрывать вентильные схемы; описание того, какую вентильную схему и когда открывать, составляет программу (микропрограмму) для машины, система команд которой включает команду «Открыть вентильную схему».

Логическое умножение осуществляется с помощью вентильной схемы, на входы которой подаются коды операндов, а на выходе образуется искомый результат (см. параграфы 1.2 и 2.5).

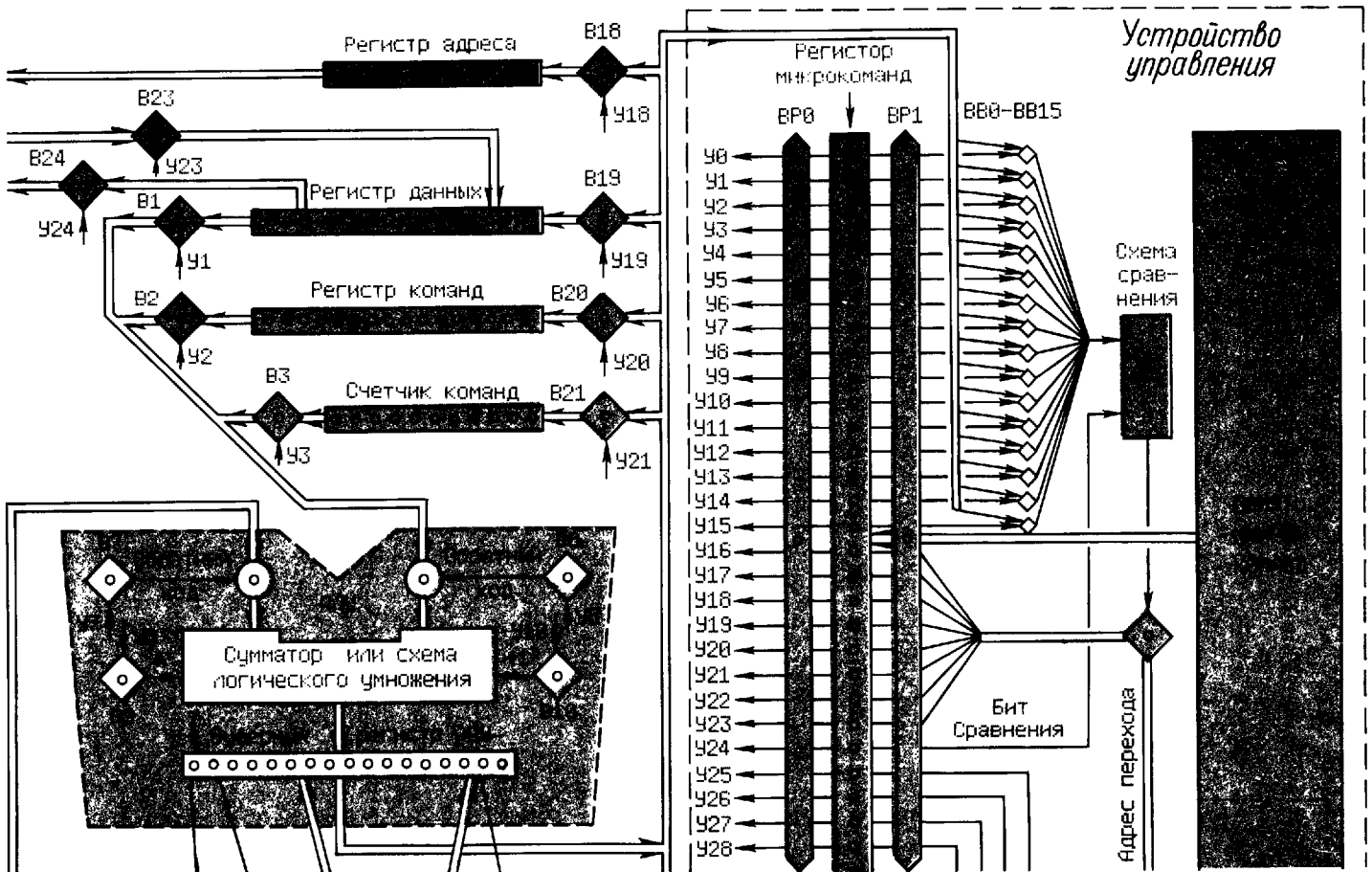
В АЛУ сумматор и схема логического умножения объединены в один блок. В блоке входные шины разветвляются на оба эти устройства, а выходы устройств через вентильные схемы подключены ко входам буферного регистра. Следовательно, при поступлении операндов одновременно выполняются операции суммирования и логического умножения, но так как по сигналу с *B9* (см. рис. 4.5 и 4.7) открыта лишь одна из выходных вентильных схем, то на буферный регистр поступит только один из двух результатов. Управление выходными вентильными схемами осуществляется подобно тому, как управляются *ВИ1* и *ВИ2* на рис. 4.6: при отсутствии управляющего сигнала на *B9* в БР записывается сумма, а при наличии управляющего сигнала на *B9* — результат логического умножения.

Циклический сдвиг на один разряд вправо или влево производится путем подачи единичного управляющего сигнала на вентильную схему *B11* или *B12* соответственно, а затем на вентильные схемы *B13* и *B22* (см. рис. 4.4 и 4.5).

Установка признаков переноса из старшего разряда сумматора, а также отрицательного или нулевого значения результата осуществляется с помощью посылки управляющих сигналов на вентили *B13*, *B14* и *B15* соответственно.

При подаче управляющего сигнала на вентиль *B14* выполняется перепись содержимого 15-го (знакового) разряда БР в однобитовый регистр *N*. При открывании вентиля *B15* производится перепись в однобитовый регистр *Z* содержимого специальной схемы, выходной сигнал которой равен единице только тогда, когда во всех 16 разрядах буферного регистра (с 0-го по 15-й) содержатся нули, т. е. когда в БР хранится код числа 0.

Содержание регистра переноса также может быть изменено с помощью управляющих сигналов. Подача сигнала на *B13* позволяет переписать в *C* содержимое старшего разряда БР (разряда с номером 16), в котором хранится перенос из старшего разряда сумматора. Посылка управляющего сигнала на *B16* приводит к сбросу *C* (команда CLC — очистка регистра переноса), а подача сигнала на *B17* — к записи в него единицы.


Устройство управления
Регистр адреса
В18
У18
В23
У23
В24
У24
Регистр данных
В19
У19
В1
У1
Регистр команд
В20
У20
В2
У2
Счетчик команд
В21
У21
В3
У3
Сумматор или схема логического умножения
Регистор микрокоманд
ВР0
ВР1
ВВ0-ВВ15
У0
У1
У2
У3
У4
У5
У6
У7
У8
У9
У10
У11
У12
У13
У14
У15
У16
У17
У18
У19
У20
У21
У22
У23
У24
У25
У26
У27
У28
Схема сравнения
Бит Сравнения
Адрес перехода

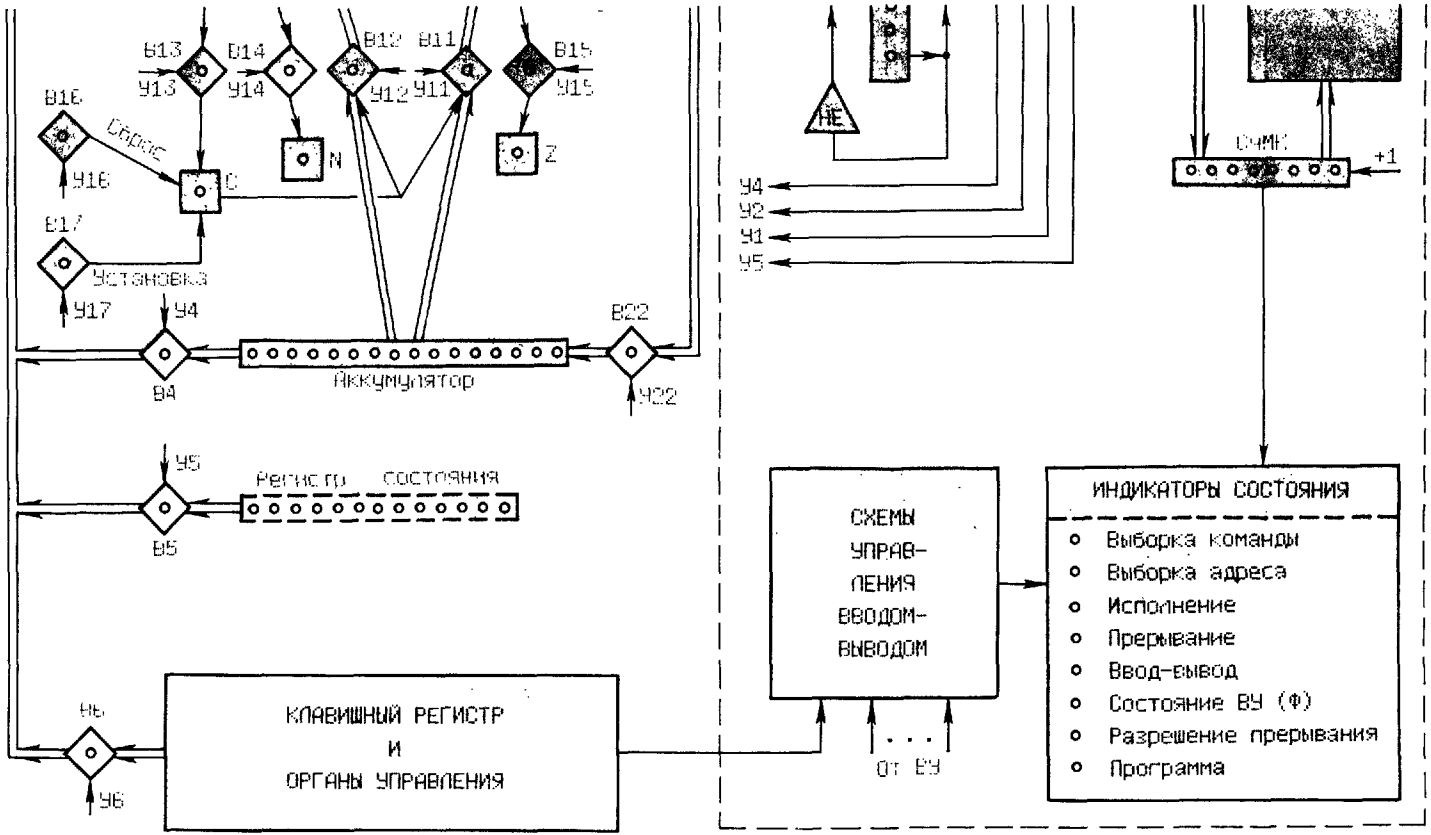


Рис. 4.7. Второй отладочный пульт базовой ЭВМ

Таблица 4.2

Регистр состояния

Разряд	Содержимое
0	Перенос (*C*)
1	Нуль (*Z*)
2	Знак (*N*)
3	0 – используется для организации безусловных переходов в МПУ
4	Разрешение прерывания
5	Прерывание
6	Состояния ВУ (Ф)
7	Состояние тумблера РАБОТА/ОСТАНОВ (1 – РАБОТА)
8	Программа
9	Выборка команды
10	Выборка адреса
11	Исполнение
12	Ввод-вывод

Регистр состояний. Это объединение однобитовых регистров признаков и состояний, назначение которых было рассмотрено здесь и в параграфе 3.3, сделано в целях формального уменьшения числа регистров, с которыми работает микропрограммное устройство управления, что позволяет сократить разрядность управляющих микрокоманд.

На втором отладочном пульте базовой ЭВМ (см. рис. 4.7) регистр состояний (РС) изображен (для удобства работы) в виде самостоятельного регистра. На самом деле его состояния лишь дублируют состояния однобитовых регистров *C*, *Z*, *N* и т. д. Соответствие между этими регистрами и разрядами РС приведено в табл. 4.2.

Вентильные схемы. Ниже перечислены функции ряда вентильных схем, используемых для пересылки информации между регистрами, между регистрами и АЛУ и для управления микрооперациями в АЛУ. Функции остальных вентильных схем на рис. 4.7 будут рассмотрены в параграфе 4.3.

Отметим, что обмен данными между регистрами, имеющими разную длину (например, между БР и СК), выполняется с помощью прямой передачи (см. рис. 4.3) и старшие разряды более длинного приемника заполняются нулями.

Входные сигналы АЛУ. Вентильные схемы В1, В2 и В3 предназначены для правого входа АЛУ, а В4, В5 и В6 – для левого. Все они используются для пересылки содержимого соответствующих регистров в буферный регистр АЛУ и далее в микропрограммное устройство управления.

В1. РД в АЛУ. Используется при арифметических и логической операциях, а также при передачах через АЛУ в РК, СК, РА и МПУ.

В2. РК в АЛУ. Используется для передачи содержимого РК в МПУ.

В3. СК в АЛУ. Используется для прибавления 1 к содержимому СК и передачи через АЛУ в РА, РД и МПУ.

В4. А в АЛУ. Используется при арифметических и логической операциях, а также при передачах через АЛУ в РД и МПУ.

В5. РС в АЛУ. Используется для передачи содержимого РС в МПУ.

В6. КР в АЛУ. Используется во время работы с пультом управления ЭВМ для передачи содержимого клавишного регистра (КР) в СК и РД.

Арифметические операции и сдвиги.

В7. Когда эта вентильная схема открыта, левый входной сигнал сумматора поступает в обратном коде.

В8. Когда эта вентильная схема открыта, правый входной сигнал сумматора поступает в обратном коде.

В9. Когда эта вентильная схема открыта, на БР поступает результат логического умножения операндов. При закрытой схеме на БР поступает сумма операндов.

В10. Когда эта вентильная схема открыта, к сумме операндов добавляется 1.

В11. Когда эта вентильная схема открыта, содержимое А и С асимметрично пересылается в БР: содержимое младшего разряда А попадает в

16-й бит БР, содержимое С — в 15-й бит БР, содержимое старшего разряда А — в 14-й бит БР и т. д.

B12. Когда эта вентильная схема открыта, содержимое А и С асимметрично пересылается в БР: содержимое младшего разряда А попадает в 1-й бит БР, содержимое С — в 0-й бит БР, содержимое старшего разряда А — в 16-й бит БР и т. д.

Установка признаков.

B13. Когда эта вентильная схема открыта, перенос (0 или 1) из старшего разряда сумматора, зафиксированный в старшем разряде БР, поступает в С.

B14. Когда эта вентильная схема открыта, знак числа, хранящегося на БР, поступает в N.

B15. Когда эта вентильная схема открыта, в Z поступает 1, если содержимое БР равно 0, и 0, если содержимое БР не равно 0.

B16. Когда эта вентильная схема открыта, в С записывается 0.

B17. Когда эта вентильная схема открыта, в С записывается 1.

Выходной сигнал АЛУ (БР).

B18. БР в РА. Используется при передаче в РА содержимого РД или СК.

B19. БР в РД. Используется при передаче в РД содержимого СК, А или КР.

B20. БР в РК. Используется при передаче в РК содержимого РД.

B21. БР в СК. Используется при передаче в СК содержимого РД или КР и для наращивания СК на 1.

B22. БР в А. Используется для засылки результатов арифметических операций и сдвигов в А.

Чтение из памяти интерпретируемой машины и запись в эту память.

B23. Из памяти в РД. Используется для загрузки в РД слова памяти, адрес которого указан в РА.

B24. Из РД в память. Используется для записи содержимого РД в слово памяти, адрес которого указан в РА.

Организация ввода-вывода информации.

B25. Когда эта вентильная схема открыта, в контроллеры ВУ пересылаются из РД приказ на ввод-вывод и адрес требуемого ВУ.

B26. Когда эта вентильная схема открыта, на все контроллеры ВУ одновременно поступает сигнал, сбрасывающий их флаги.

B27. Когда эта вентильная схема открыта, производится сброс регистров прерывания и разрешения прерывания.

B28. Когда эта вентильная схема открыта, производится установка единицы в регистр разрешения прерывания.

Останов ЭВМ.

B0. Эта вентильная схема используется для передачи сигнала прекращения выполнения программы (команда HLT).

Другие компоненты процессора (память микрокоманд, регистр микрокоманд и т. д.) рассмотрены в параграфе 4.3.

Нетрудно заметить, что в предложенной структуре передача данных между регистрами осуществляется только через буферный регистр АЛУ. Это сделано для уменьшения числа шин и вентилей, т. е. для упрощения и удешевления процессора. Однако такое решение замедляет процедуру обмена.

4.3. Микропрограммное управление вентильными схемами

Микрокоманды. Интерпретирующий уровень (см. рис. 4.7) обладает набором команд, состоящим только из двух микрокоманд: операционной (ОМК) и управляющей (УМК), форматы которых показаны на рис. 4.8.

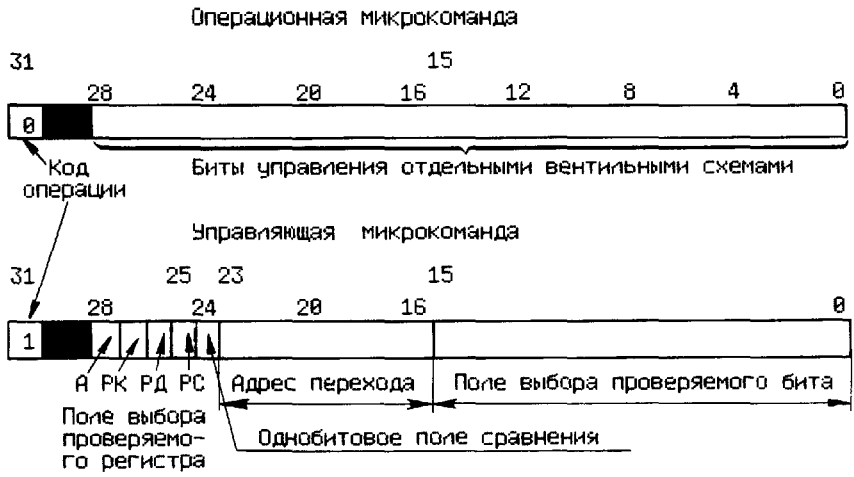


Рис. 4.8. Система команд микропрограммного уровня, показанного на рис. 4.7 (состоит только из двух команд)

Поскольку команд только две, достаточно однобитового кода операции: 0 — для ОМК и 1 — для УМК. Хотя для размещения информации и кода операции достаточно 30 бит, были выбраны 32-разрядные (четырехбайтовые) микрокоманды.

В операционной микрокоманде каждый из 29 информационных битов управляет одной из 29 вентильных схем, описанных в параграфе 4.2. Единица означает, что вентильная схема должна быть открыта, нуль — закрыта. Подобная структура микрокоманды обычно называется горизонтальной.

Управляющая микрокоманда используется для изменения последовательности выполнения микрокоманд в зависимости от тех или иных условий. Каждая микрокоманда УМК определяет 1 бит, подлежащий проверке (любой из 61 бит, находящихся в *А*, *РК*, *РД* и *РС*). Если выбранный бит и 24-й бит микрокоманды совпадают, последовательность микрокоманд, подлежащих исполнению, изменяется и адрес следующей микрокоманды определяется битами с 16 по 23 самой УМК. Если выбранный бит отличается от бита 24, последовательность выполнения микрокоманд не претерпевает изменений.

Бит, подлежащий проверке, определяется двумя полями микрокоманды. Биты с 25 по 28 указывают, какой регистр интерпретирующей машины должен быть проверен согласно следующим правилам:

бит 25 — регистр состояний (РС);
бит 26 — регистр данных (РД);
бит 27 — регистр команд (РК);
бит 28 — аккумулятор (А).

Биты микрокоманды с 0 по 15 указывают, какой бит выбранного регистра подлежит проверке. Только один бит, принадлежащий группе битов с 0 по 15, и один бит группы битов с 25 по 28 должны быть равны единице.

Например, после увеличения на единицу содержимого регистра данных (РД) в команде ISZ (см. параграф 2.4) надо проверить его знаковый разряд (разряд с номером 15). Если этот разряд равен 1 (содержимое РД меньше нуля), то выполнение микрокоманд команды ISZ завершается. В противном случае необходимо прибавить единицу к содержимому счетчика команд, т. е. организовать пропуск следующей за ISZ команды. Разветвление в микропрограмме реализации команды ISZ (переход по адресу 8F) осуще-

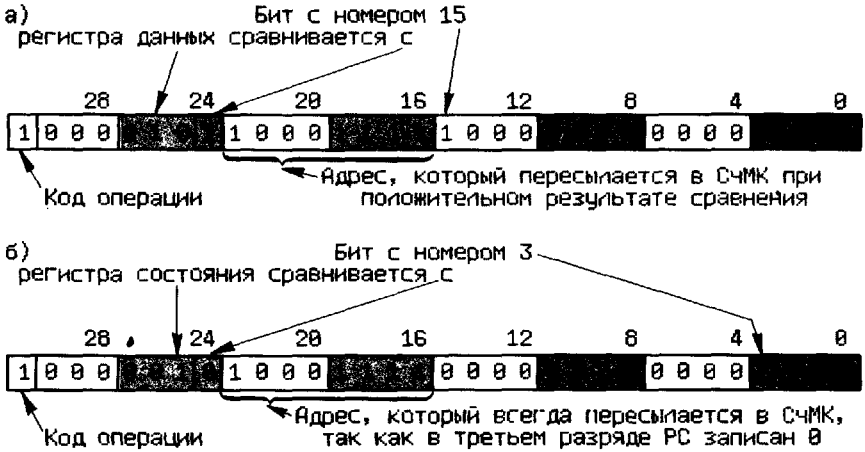


Рис. 4.9. Примеры управляющих микрокоманд:
а — условного перехода; *б* — безусловного перехода

ствляется с помощью микрокоманды $(858F8000)_{16}$. На рис. 4.9, *а* эта микрокоманда представлена в двоичной форме с указанием всех используемых битов.

Для организации безусловного перехода (например, по тому же адресу 8F) используется 3-й бит регистра состояний, содержащий константу 0. Сравнение этого разряда с константой 0, установленной в 24-й разряд УМК, всегда дает положительный результат и позволяет переслать в СчМК нужный адрес перехода (рис. 4.9, *б*).

Выполнение микропрограмм. Микропрограмма хранится в памяти микрокоманд, которая состоит из 256 (2^8) 32-битовых слов (см. рис. 4.7). Для управления выполнением микропрограммы используется счетчик микрокоманд (СчМК), аналогичный счетчику команд традиционного машинного уровня. Чтобы извлекать микрокоманды из памяти микрокоманд, декодировать их и исполнять, следовало бы иметь регистры адреса, данных и команд. Но поскольку машина-интерпретатор очень проста, один регистр выполнит функции регистра адреса и счетчика микрокоманд, а другой - регистра данных и регистра микрокоманд (РМК).

Работу процессора можно описать следующим образом. При появлении тактового импульса из памяти микрокоманд извлекается и загружается в РМК слово, на которое указывает СчМК, и к содержимому этого счетчика прибавляется единица. Если из памяти извлечена УМК, то в 31-м бите РМК содержится 1 (код операции УМК), которая открывает вентильную схему ВР1 и тем самым создает условия для исполнения УМК. Если же извлечена ОМК, то в 31-м бите РМК — нуль. Этот сигнал с помощью инвертора НЕ открывает вентильную схему *ВР0*, и через нее на *В0* -*В28* передаются состояния соответствующих битов РМК. Разряды РМК, содержащие единицы, создают открывающий управляющий сигнал, а содержащие нули — закрывающий (*У0* — *У28*).

При исполнении УМК по сигналу, создаваемому каким-либо битом поля выбора проверяемого регистра (*У1*, *У2*, *У4* или *У5*), открывается одна из вентильных схем *В1*, *В2*, *В4* или *В5* и на вентили *ВВ0* — *ВВ15* поступает через АЛУ содержимое соответствующего регистра. Одновременно на эти же вентили поступает с РМК содержимое поля выбора проверяемого бита. Так как в этом поле должна быть записана только одна единица, то откры-

вается лишь один из вентилей *ВВ0 — ВВ15*, через который на схему сравнения поступает содержимое проверяемого бита из проверяемого регистра.

На второй вход схемы сравнения поступает содержимое однобитового поля сравнения (24-й бит УМК), в которое при кодировании УМК записали цифру 0 или 1. Если проверяемый бит и бит из поля сравнения идентичны, то схема сравнения формирует единичный сигнал, который открывает вентильную схему ВА, и на СчМК пересылается адрес перехода (биты с 16 по 23). В противном случае на СчМК сохраняется адрес микрокоманды, расположенной вслед за исполняемой.

Когда на СчМК появляется адрес микрокоманды, с которой начинается какой-либо из машинных циклов (см. параграфы 2.5 и 3.3), изменяется информация на индикаторе состояний процессора. Эта информация изменяется и при переходе ЭВМ из состояния ОСТАНОВ в состояние ПРОГРАММА или обратном переходе.

4.4. Интерпретатор базовой ЭВМ

Язык микропрограммирования. Микропрограммы можно писать в виде последовательности 32-битовых двоичных чисел, однако такая работа весьма утомительна и можно легко допустить ошибку. Для облегчения записи и проверки микропрограмм выполнения отдельных машинных команд и циклов используем их шестнадцатеричное кодирование. Комментарии к микрокомандам будем записывать с помощью мнемонического языка микропрограммирования, напоминающего алгоритмические языки высокого уровня (БЕЙСИК, ФОРТРАН и др.). Рассмотрим примеры такой записи для различных микроопераций.

Пересылка информации из регистра в регистр (регистры) комментируется посредством оператора присваивания. Например, пересылка содержимого БР в РД (открыть В19) записывается в виде

0008 0000; БР ⇒ РД.

(Для облегчения шестнадцатеричного кодирования используйте рис. 4.9, где представлено разбиение 32-битового регистра на тетрады.)

Пересылка содержимого БР в аккумулятор с установкой признаков результата (открыть В13, В14, В15 и В22) записывается в виде

0040 E000; БР ⇒ A, C, N, Z.

Обмен информацией между памятью (ОП) и регистром данных задается следующим образом:

чтение из памяти по адресу, хранимому в РА (открыть В23),

0080 0000; ОП (РА) ⇒ РД;

запись в память по адресу, хранимому в РА (открыть В24),

0100 0000; РД ⇒ ОП (РА).

Операции, выполняемые в АЛУ, также комментируются операторами присваивания. При этом обратный код записывается как функция COM, а сдвиги вправо и влево — как функции RAR и RAL соответственно. Например, суммирование А и РД (открыть В1 и В4) запишется в виде

0000 0012; А + РД ⇒ БР,

вычитание РД из А (открыть В1, В4, В8 и В10)

0000 0512; А + COM (РД) + 1 ⇒ БР,

циклический сдвиг содержимого A и C на один разряд вправо (открыть B11)

0000 0800; RAR (A, C) ⇒ БР

и, наконец, прибавление единицы к содержимому СК (открыть B3 и B10)

0000 0408; СК+1 ⇒ БР.

Управляющая микрокоманда комментируется с помощью оператора условного перехода

IF BIT (к, рег) = B THEN метка (адрес),

где к — номер бита, подлежащего проверке (от 0 до 15); рег — регистр, подлежащий проверке (A, РК, РД, РС); B — 0 или 1; метка (адрес) — метка и адрес микрокоманды, которой передается управление, если проверка дает положительный результат.

Например, проверка бита 12 регистра команд запишется так:

89aa 1000; IF BIT (12, РК) = 1 THEN E1 (aa),

где вместо «aa» должен быть проставлен адрес микрокоманды, которой передается управление при положительном результате проверки; комментарии этой микрокоманды должны иметь метку E1.

Проверка 15-го бита аккумулятора запишется в виде

90aa 8000; IF BIT (15, A) = 0 THEN HE (aa).

Напомним, что проверка 3-го разряда регистра состояний, содержащего константу 0, с целью определения, равен ли он нулю, всегда дает положительный результат. Это порождает безусловную передачу управления, которая записывается в виде

82aa 0008; GO TO BR (aa).

Микропрограмма. Полный текст интерпретатора приведен в табл. 4.3. В этой таблице есть одна «лишняя» графа (ВЕРТИКАЛЬНАЯ), содержимое которой описано в параграфе 4.5.

Строки микропрограммы с номерами 01—0C описывают процессы выборки команды интерпретируемой машины, пересылки ее в РК для анализа и увеличения содержимого СК на единицу. Строки с номерами 07—0B образуют декодирующее «дерево», которое проверяет биты 15, 14, 13 и 12 регистра команд для определения типа команды и передачи управления в соответствующую часть микропрограммы. При этом используется то обстоятельство, что коды операций адресных команд расположены в диапазоне от 0000 до 1100, т. е. обязательно содержат цифру 0 в 15, 14 и 13 разрядах, а команды ввода-вывода (код операции 1110) отличаются от безадресных команд (код операции 1111) 12 разрядом.

Если выбрана адресная команда, то в строке 0C проверяется заданный в ней вид адресации (прямая или косвенная). При прямой адресации осуществляется переход на строку 1D, а при косвенной — переход к микрокомандам цикла выборки адреса операнда (строки 0D — 1C) и далее к строке 1D.

В цикле выборки адреса операнда сначала производится выборка в РД этого адреса, а затем — проверка: не хранился ли он в одной из индексных ячеек (ячеек с адресами от 008 до 00F). Если да, то содержимое РД увеличивается на 1, затем пересылается обратно в индексную ячейку и, наконец, восстанавливается (от содержимого РД вычитается 1). Далее начинают выполняться микрокоманды цикла исполнения адресных команд.

Таблица 4.3

Интерпретатор учебной ЭВМ (микропрограмма)

АД-РЕС	МИКРОКОМАНДЫ: ГОРИЗОНТАЛЬНАЯ	МИКРОКОМАНДЫ: ВЕРТИКАЛЬНАЯ	МИКРОКОМАНДЫ: МЕТКА	КОММЕНТАРИИ: ДЕЙСТВИЕ
1	2	3	4	5
		ЦИКЛ ВЫБОРКИ КОМАНДЫ		
01:	0000 0008	0300	НАЧ	СК => БР
02:	0004 0000	4001		БР => РА
03:	0080 0408	0311		ОП(РА) => РД, СК+1 => БР
04:	0020 0000	4004		БР => СК
05:	0000 0002	0100		РД => БР
06:	0010 0000	4003		БР => РК
				ОПРЕДЕЛЕНИЕ ТИПА КОМАНДЫ
07:	880C 8000	AF0C		IF BIT(15,РК)=0 THEN АДЧ(0C)
08:	880C 4000	AE0C		IF BIT(14,РК)=0 THEN АДЧ(0C)
09:	880C 2000	AD0C		IF BIT(13,РК)=0 THEN АДЧ(0C)
0A:	895E 1000	EC5E		IF BIT(12,РК)=1 THEN БАД(5E)
0B:	828E 0008	838E		GOTO В/В(8E)
				ОПРЕДЕЛЕНИЕ ВИДА АДРЕСАЦИИ
0C:	881D 0800	AB1D	АДЧ	IF BIT(11,РК)=0 THEN АДР(1D)
		ЦИКЛ ВЫБОРКИ АДРЕСА ОПЕРАНДА		
0D:	0000 0002	0100		РД => БР
0E:	0004 0000	4001		БР => РА
0F:	0080 0000	0001		ОП(РА) => РД
10:	881D 0008	A31D		IF BIT(3,РК)=0 THEN АДР(1D)
11:	891D 0010	E41D		IF BIT(4,РК)=1 THEN АДР(1D)
12:	891D 0020	E51D		IF BIT(5,РК)=1 THEN АДР(1D)
13:	891D 0040	E61D		IF BIT(6,РК)=1 THEN АДР(1D)
14:	891D 0080	E71D		IF BIT(7,РК)=1 THEN АДР(1D)
15:	891D 0100	E81D		IF BIT(8,РК)=1 THEN АДР(1D)
16:	891D 0200	E91D		IF BIT(9,РК)=1 THEN АДР(1D)
17:	891D 0400	EA1D		IF BIT(10,РК)=1 THEN АДР(1D)
18:	0000 0402	0110		РД+1 => БР
19:	0008 0000	4002		БР => РД
1A:	0100 0000	0002		РД => ОП(РА)
1B:	0000 0082	0140		РД+СОМ(0)=РД-1 => БР
1C:	0008 0000	4002		БР => РД
		ЦИКЛ ИСПОЛНЕНИЯ АДРЕСНЫХ КОМАНД		
				ДЕКОДИРОВАНИЕ АДРЕСНЫХ КОМАНД
1D:	892D 8000	EF2D	АДР	IF BIT(15,РК)=1 THEN ПРХ(2D)
1E:	0000 0002	0100		РД => БР
1F:	0004 0000	4001		БР => РА
20:	8927 4000	EE27		IF BIT(14,РК)=1 THEN АРФ(27)
21:	8824 2000	AD24		IF BIT(13,РК)=0 THEN A1(24)
22:	8857 1000	AC57		IF BIT(12,РК)=0 THEN JSR(57)
23:	8238 0008	8338		GOTO MOV(38)
24:	0080 0000	0001	A1	ОП(РА) => РД
25:	8850 1000	AC50		IF BIT(12,РК)=0 THEN ISZ(50)
26:	8235 0008	8335		GOTO AND(35)
27:	0080 0000	0001	АРФ	ОП(РА) => РД
28:	882B 2000	AD2B		IF BIT(13,РК)=0 THEN СУМ(2B)

1	2	3	4	5
29:	8843 1000	AC43		IF BIT(12,РК)=0 THEN SUB(43)
2A:	82B0 0008	83B0		GOTO Р-А(В0)
2B:	883C 1000	AC3C	СУМ	IF BIT(12,РК)=0 THEN ADD(3C)
2C:	823F 0008	833F		GOTO ADC(3F)
2D:	8830 4000	AE30	ПРХ	IF BIT(14,РК)=0 THEN УПХ(30)
2E:	8847 1000	AC47		IF BIT(12,РК)=0 THEN BR(47)
2F:	82D0 0008	83D0		GOTO Р-П(D0)
30:	8833 2000	AD33	УПХ	IF BIT(13,РК)=0 THEN П1(33)
31:	884C 1000	AC4C		IF BIT(12,РК)=0 THEN BMI(4C)
32:	824E 0008	834E		GOTO BEQ(4E)
33:	8846 1000	AC46	П1	IF BIT(12,РК)=0 THEN BCS(46)
34:	824A 0008	834A		GOTO BPL(4A)
				ИСПОЛНЕНИЕ АДРЕСНЫХ КОМАНД
35:	0000 0212	1120	AND	А & РД => БР
36:	0040 C000	4035		БР => A,N,Z
37:	828F 0008	838F		GOTO ПРЕ(8F)
38:	0000 0010	1000	MOV	А => БР
39:	0008 0000	4002		БР => РД
3A:	0100 0000	0002		РД => ОП(РА)
3B:	828F 0008	838F		GOTO ПРЕ(8F)
3C:	0000 0012	1100	ADD	А + РД => БР
3D:	0040 E000	4075		БР => A,C,N,Z
3E:	828F 0008	838F		GOTO ПРЕ(8F)
3F:	823C 0001	803C	ADC	IF BIT(0,PC)=0 THEN ADD(3C)
40:	0000 0412	1110		А + РД + 1 => БР
41:	0040 E000	4075		БР => A,C,N,Z
42:	828F 0008	838F		GOTO ПРЕ(8F)
43:	0000 0512	1190	SUB	А+COM(РД)+1=А-РД => БР
44:	0040 E000	4075		БР => A,C,N,Z
45:	828F 0008	838F		GOTO ПРЕ(8F)
46:	828F 0001	808F	BCS	IF BIT(0,PC)=0 THEN ПРЕ(8D)
47:	0000 0002	0100	BR	РД => БР
48:	0020 0000	4004		БР => СК
49:	828F 0008	838F		GOTO ПРЕ(8F)
4A:	838F 0004	C28F	BPL	IF BIT(2,PC)=1 THEN ПРЕ(8F)
4B:	8247 0008	8347		GOTO BR(47)
4C:	828F 0004	828F	BMI	IF BIT(2,PC)=0 THEN ПРЕ(8F)
4D:	8247 0008	8347		GOTO BR(47)
4E:	828F 0002	818F	BEQ	IF BIT(1,PC)=0 THEN ПРЕ(8F)
4F:	8247 0008	8347		GOTO BR(47)
50:	0000 0402	0110	ISZ	РД + 1 => БР
51:	0008 0000	4002		БР => РД
52:	0100 0000	0002		РД => ОП(РА)
53:	858F 8000	DF8F		IF BIT(15,РД)=1 THEN ПРЕ(8F)
54:	0000 0408	0310		СК + 1 => БР
55:	0020 0000	4004		БР => СК
56:	828F 0008	838F		GOTO ПРЕ(8F)
57:	0000 0402	0110	JSR	РД + 1 => БР
58:	0010 0000	4003		БР => РК
59:	0000 0008	0300		СК => БР
5A:	0008 0000	4002		БР => РД
5B:	0100 0004	0202		РД => ОП(РА), РК => БР
5C:	0020 0000	4004		БР => СК
5D:	828F 0008	838F		GOTO ПРЕ(8F)

1	2	3	4	5
				ПРОДОЛЖЕНИЕ ЦИКЛА ВЫБОРКИ КОМАНДЫ ДЕКОДИРОВАНИЕ И ИСПОЛНЕНИЕ БЕЗАДРЕСНЫХ КОМАНД
5E:	8861 0800	AB61	БАД	IF BIT(11,PK)=0 THEN Б0(61)
5F:	886C 0400	AA6C		IF BIT(10,PK)=0 THEN Б1(6C)
60:	82E0 0008	83E0		GOTO Р-Б(E0)
61:	8867 0400	AA67	Б0	IF BIT(10,PK)=0 THEN Б2(67)
62:	8865 0200	A965		IF BIT(9,PK)=0 THEN Б3(65)
63:	8882 0100	A882		IF BIT(8,PK)=0 THEN ROL(82)
64:	8285 0008	8385		GOTO ROR(85)
65:	887B 0100	A87B	Б3	IF BIT(8,PK)=0 THEN CMA(7B)
66:	827E 0008	837E		GOTO CMC(7E)
67:	886A 0200	A96A	Б2	IF BIT(9,PK)=0 THEN Б4(6A)
68:	8876 0100	A876		IF BIT(8,PK)=0 THEN CLA(76)
69:	8279 0008	8379		GOTO CLC(79)
6A:	8888 0100	A888	Б4	IF BIT(8,PK)=0 THEN HLT(88)
6B:	8287 0008	8387		GOTO NOP(87)
6C:	886F 0200	A96F	Б1	IF BIT(9,PK)=0 THEN Б5(6F)
6D:	888A 0100	A88A		IF BIT(8,PK)=0 THEN EI(8A)
6E:	828C 0008	838C		GOTO DI(8C)
6F:	8873 0100	A873	Б5	IF BIT(8,PK)=0 THEN INC(73)
70:	0000 0110	1080	DEC	A+COM(0)=A-1 => БР
71:	0040 E000	4075		БР => A,C,N,Z
72:	828F 0008	838F		GOTO ПРЕ(8F)
73:	0000 0410	1010	INC	A + 1 => БР
74:	0040 E000	4075		БР => A,C,N,Z
75:	828F 0008	838F		GOTO ПРЕ(8F)
76:	0000 0200	0020	CLA	0 => БР
77:	0040 C000	4035		БР => A,N,Z
78:	828F 0008	838F		GOTO ПРЕ(8F)
79:	0001 0000	408C	CLC	0 => C
7A:	828F 0008	838F		GOTO ПРЕ(8F)
7B:	0000 009C	1040	CMA	COM(A) => БР; ИНВЕРСИЯ A
7C:	0040 C000	4035		БР => A,N,Z
7D:	828F 0008	838F		GOTO ПРЕ(8F)
7E:	8280 0001	8000	CMC	IF BIT(0,PC)=0 THEN Б6(80)
7F:	8279 0008	8379		GOTO CLC(79)
80:	0002 0000	4000	Б6	1 => C
81:	828F 0008	838F		GOTO ПРЕ(8F)
82:	0000 1000	0008	ROL	RAL(A) => БР; СДВИГ ВЛЕВО
83:	0040 E000	4075		БР => A,C,N,Z
84:	828F 0008	838F		GOTO ПРЕ(8F)
85:	0000 0800	0004	ROR	RAR(A)=> БР; СДВИГ ВПРАВО
86:	0040 E000	4075		БР => A,C,N,Z
87:	828F 0008	838F	NOP	GOTO ПРЕ(8F)
88:	0000 0001	4008	HLT	ОСТАНОВ МАШИНЫ
89:	8201 0008	8301		GOTO НАЧ(01)
8A:	1000 0000	4800	EI	РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ
8B:	8201 0008	8301		GOTO НАЧ(01)
8C:	0500 0000	4400	DI	ЗАПРЕЩЕНИЕ ПРЕРЫВАНИЯ
8D:	8201 0008	8301		GOTO НАЧ(01)

```
-----------------------------------------------------------------
: 1 :      2     :  3   :  4  :              5                  :
:---------------------------------------------------------------:
:                   ПРОДОЛЖЕНИЕ ЦИКЛА ВЫБОРКИ КОМАНДЫ           :
:                      ДЕКОДИРОВАНИЕ И ИСПОЛНЕНИЕ               :
:                          КОМАНД ВВОДА-ВЫВОДА                  :
: 8E: 0200 0000 : 4100 : В/В :    ОРГАНИЗАЦИЯ СВЯЗЕЙ С ВУ       :
:                  Ц И К Л    П Р Е Р Ы В А Н И Я               :
: 8F: 8288 0080 : 8788 : ПРЕ : IF BIT(7,PC)=0  THEN HLT(88)      :
: 90: 8201 0020 : 8501 :     : IF BIT(5,PC)=0  THEN НАЧ(01)      :
: 91: 0000 0200 : 0020 :     :    0 => БР                        :
: 92: 0004 0000 : 4001 :     :    БР => РА                       :
: 93: 0000 0008 : 0300 :     :    СК => БР                       :
: 94: 0008 0000 : 4002 :     :    БР => РД                       :
: 95: 0100 0400 : 0012 :     :    РД => ОП(РА), 1 => БР          :
: 96: 0020 0000 : 4004 :     :    БР => СК                       :
: 97: 0800 0000 : 4400 :     :    ЗАПРЕЩЕНИЕ ПРЕРЫВАНИЯ          :
: 98: 8201 0008 : 8301 :     :                 GOTO НАЧ(01)       :
:                  П У Л Ь Т О В Ы Е   О П Е Р А Ц И И           :
:                                В В О Д    А Д Р Е С А          :
: 99: 0000 0040 : 3000 : В/А :    КР => БР                       :
: 9A: 0020 0000 : 4004 :     :    БР => СК                       :
: 9B: 828F 0008 : 838F :     :                 GOTO ПРЕ(8F)       :
:                                  Ч  Т  Е  Н  И  Е              :
: 9C: 0000 0008 : 0300 :  ЧТ :    СК => БР                       :
: 9D: 0004 0000 : 4001 :     :    БР => РА                       :
: 9E: 0080 0408 : 0311 :     :    ОП(РА) => РД, СК+1 => БР       :
: 9F: 0020 0000 : 4004 :     :    БР => СК                       :
: A0: 828F 0008 : 838F :     :                 GOTO ПРЕ(8F)       :
:                                  З  А  П  И  С  Ь              :
: A1: 0000 0008 : 0300 : ЗАП :    СК => БР                       :
: A2: 0004 0000 : 4001 :     :    БР => РА                       :
: A3: 0000 0040 : 3000 :     :    КР => БР                       :
: A4: 0008 0000 : 4002 :     :    БР => РД                       :
: A5: 0100 0408 : 0312 :     :    РД => ОП(РА), СК+1 => БР       :
: A6: 0020 0000 : 4004 :     :    БР => СК                       :
: A7: 828F 0008 : 838F :     :                 GOTO ПРЕ(8F)       :
:                                   П   У   С   К                :
: A8: 0000 0200 : 0020 : ПУС :    0 => БР                        :
: A9: 005C E000 : 4077 :     :    БР => A,C,N,Z,РА,РД,РК         :
: AA: 0400 0000 : 4200 :     :    СБРОС ФЛАГОВ ВУ                :
: AB: 0800 0000 : 4400 :     :    ЗАПРЕЩЕНИЕ ПРЕРЫВАНИЯ          :
: AC: 828F 0008 : 838F :     :                 GOTO ПРЕ(8F)       :
:                     . . .                                      :
: B0:                 : Р-А : АРИФМЕТИЧЕСКАЯ КОМАНДА   7###      :
:                     . . .                                      :
: D0:                 : Р-П :    КОМАНДА  ПЕРЕХОДА     D###     :
:                     . . .                                      :
: E0:                 : Р-Б :   БЕЗАДРЕСНАЯ КОМАНДА    FC##      :
:                     . . .                                      :
: FF:                                                            :
-----------------------------------------------------------------
```

В цикле исполнения адресных команд, который начинается с микрокоманды отделения команд переходов, расположенной по адресу 1D, производятся дальнейшее декодирование команды и ее исполнение. После отделения команд переходов осуществляется отделение арифметических команд (SUB, ADD, ADC), команд ISZ и AND и т. д.

Реализация любой адресной команды (впрочем, как и любой другой команды) завершается микрокомандой перехода к циклу прерывания (микрокоманды 8F – 98), где выясняется режим работы ЭВМ и, если машина находится в состоянии «ПРОГРАММА», то проверяется, не требуется ли прервать выполнение программы по запросу внешнего устройства (см. параграф 3.5). Если не требуется, то осуществляется переход к началу микропрограммы (строка 01), т. е. к выборке новой команды интерпретируемой машины. В противном случае организуется переход к подпрограмме обработки прерываний.

Аналогичным образом обрабатываются безадресные команды, команды ввода-вывода и команды, задаваемые нажатием кнопок на пульте управления ЭВМ.

В системе команд базовой ЭВМ ряд кодов операций зарезервирован для включения новых команд. Это арифметическая команда 7xxx, команда перехода Dxxx и безадресные команды FC00, FD00, FE00 или FF00. Когда при декодировании команды выясняется, что выбрана команда 7xxx, производится передача управления к строке B0. Следовательно, часть микропрограммы, описывающая последовательность микроопераций по реализации этой команды, должна начинаться со строки B0. Сюда можно записать, например, действия по умножению или делению операндов. Часть микропрограммы, реализующая дополнительные безадресные команды, начинается от строки E0. Первые микрокоманды этой части должны осуществить декодирование выбранной команды (определить расширение кода операции: C, D, E или F) и передать управление соответствующим микрокомандам нового куска микропрограммы. Так как память микрокоманд имеет 256 ячеек, то при составлении микропрограмм новых команд можно использовать лишь строки с номерами от AD до FF.

Заключение

Рассмотрим, наконец, что же, собственно говоря, мы называем микропрограммированием. Основная идея изложенного выше подхода сводится к тому, чтобы начать с очень простой аппаратно реализованной машины (микропрограммный уровень базовой ЭВМ имеет лишь две простые команды) и использовать ее для написания интерпретатора более сложной, так называемой интерпретируемой машины. Эта машина, в свою очередь, может выполнять роль интерпретирующей для другой виртуальной (кажущейся) машины и т. д.

4.5. Другие варианты построения микрокоманд

В параграфе 1.5 объяснялось, почему короткие команды лучше, чем длинные: такие команды позволяют сэкономить память и увеличить количество команд, которое может быть извлечено из памяти за одну секунду. Это справедливо и в отношении микропрограммного уровня. Микрокоманда с одним битом, приходящимся на одну вентильную схему (горизонтальная микрокоманда), привлекательна простотой построения и реализации (единицы и нули, содержащиеся в микрокоманде, непосредственно поступают на вентильные схемы). Однако в случае вычислительной машины сравнительно больших размеров число вентильных схем может составлять несколько сотен и горизонтальная микрокоманда окажется неприемлемо длинной.

Кодируемые поля. Не все возможные комбинации 32 бит можно использовать в рассмотренных ранее микрокомандах базовой ЭВМ (см. табл. 4.2). Так, вентильные схемы 1, 2 и 3; 1 и 19; 23 и 24 и много других групп не могут

быть открыты одновременно (см. рис. 4.7). Хотя принципиально нет ограничений на то, чтобы вентильные схемы 18, 19, 20, 21 и 22 были открыты одновременно, в действительности редко возникает необходимость в открытии в данный момент времени более одной из них.

Недопустимость более одного сигнала на левом и правом входах АЛУ, задания сигналов на входы АЛУ и использование в том же такте полученного результата, одновременной записи в БР суммы операндов и результатов их логического умножения — все это позволяет разбить операционную микрокоманду на две и сократить общее число используемых разрядов микрокоманды.

В управляющей микрокоманде сокращение ее разрядов можно обеспечить за счет кодирования полей выбора проверяемого регистра и проверяемого бита в этом регистре, что позволяет сократить УМК до 16 бит, т. е. вдвое.

Один из возможных вариантов 16-разрядных микрокоманд базовой ЭВМ приведен на рис. 4.10. Здесь почти вся информация закодирована. Например, поле выбора проверяемого бита новой УМК в четыре раза меньше, чем в старой УМК (см. рис. 4.8). Несколько сократились и другие поля (чем длиннее поле, тем больше экономия от кодирования). Однако для декодирования таких сжатых полей требуется использовать специальную схему — дешифратор.

Дешифратор «1 из 2^n» представляет собой комбинационную схему с n входами и 2^n выходами (на рис. 4.11 показаны два дешифратора: «1 из 16», имеющий 4 входа и $2^4=16$ выходов, и «1 из 4», имеющий 2 входа и $2^2=4$ выхода). Каждая выходная линия дешифратора однозначно соответствует одной из 2^n возможных комбинаций входных сигналов.

На рис. 4.11 приведена часть одного из вариантов микропрограммного устройства управления базовой ЭВМ в момент обработки 16-разрядной управляющей микрокоманды EC5E — IF BIT (12, РК) = 1 THEN БАД (5E). Значение $(10)_2$, находящееся в поле выбора проверяемого регистра (12—13 биты УМК), создает, проходя через дешифратор DC2, единичный управляющий сигнал *У2*. Этот сигнал открывает вентильную схему *В2* (см. рис. 4.7), и содержимое РК переписывается в буферный регистр (БР). Значение $(1100)_2$, находящееся в поле выбора проверяемого бита (8—11-й биты УМК), создает на 12-м выходе дешифратора DC1 единичный управляющий сигнал, который используется для выделения и пересылки на сравнивающее устройство 12-го бита БР, т. е. 12-го бита РК. Так как в однобитовом поле сравнения записана 1, то при единичном значении 12-го бита РК в счетчик микрокоманд (СчМК) переписывается адрес перехода 5E (0—7-й биты УМК).

Для декодирования операционных микрокоманд ОМК0 и ОМК1 (см. рис. 4.10) потребуются еще восемь дешифраторов. Однако экономия затрат на память микрокоманд (эта память сокращается вдвое за счет использования 16-разрядных микрокоманд) будет, вероятно, превышать стоимость этих дешифраторов.

Текст интерпретатора базовой ЭВМ, написанный с использованием 16-разрядных микрокоманд, приведен в табл. 4.3 (графа ВЕРТИКАЛЬНАЯ). Название графы обусловлено тем, что микрокоманды, состоящие из большого числа кодируемых полей, принято называть вертикальными, в отличие от горизонтальных микрокоманд, в которых каждый бит управляет одной вентильной схемой.

Разбиение такта на фазы. Объем микропрограммной памяти можно сократить и при использовании горизонтальных микрокоманд. Для этого надо увеличить их информационную емкость: кодировать в каждой микрокоман-

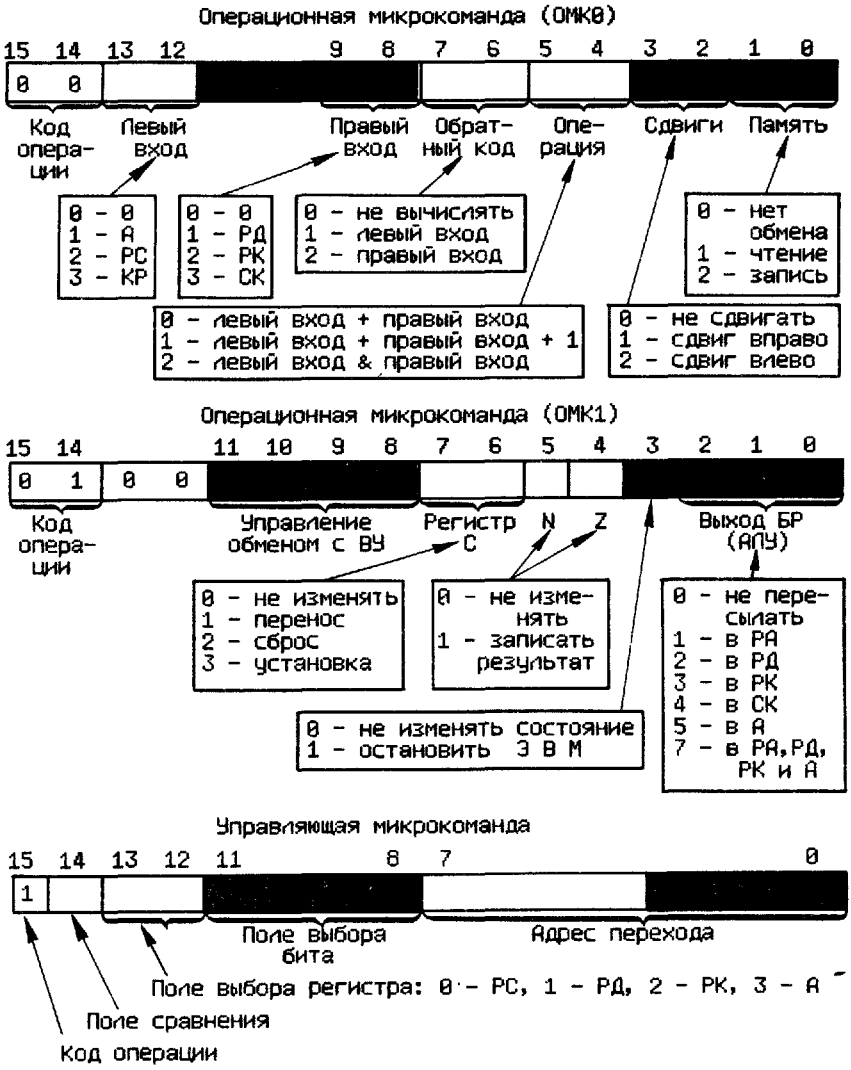


Рис. 4.10. Шестнадцатиразрядные микрокоманды базовой ЭВМ

де большее число микроопераций. Исполнение же несовместимых во времени микроопераций можно обеспечить разбиением рабочего такта ЭВМ (времени исполнения одной микрокоманды) на несколько подтактов (фаз), в каждом из которых выполняется одно или несколько элементарных действий по реализации микрокоманды.

На рис. 4.12 приведены три первые микрокоманды машинного цикла «Выборка команды» базовой ЭВМ и последовательность их выполнения при двухфазном тактировании (фазы *Ф1* и *Ф2*). По этим новым (более емким) микрокомандам выполняются те же микрооперации, что и по шести первым микрокомандам в табл. 4.3.

Нанопамять. Количество различных по содержанию горизонтальных микрокоманд обычно невелико, но часть из них используется многократно (например, РА=РК, СК=СК+1, СК=РК и другие команды базовой ЭВМ). Это обстоятельство, а также большая длина таких микрокоманд

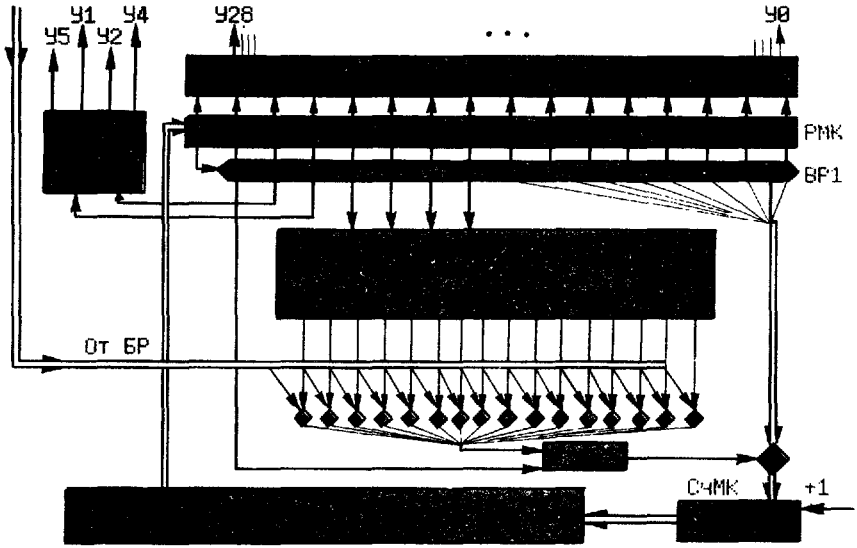


Рис. 4.11. Часть микропрограммного устройства управления с 16-разрядными микрокомандами

натолкнули разработчиков ЭВМ на мысль о размещении в специальной дополнительной памяти (нанопамяти) только неповторяющихся микрокоманд. Микропрограмма же в этом случае будет состоять лишь из управляющих микрокоманд и адресов ячеек нанопамяти.

Так как количество ячеек нанопамяти невелико, то их адреса имеют малую разрядность, что позволяет уменьшить длину ячеек памяти микрокоманд, где размещаются такие адреса. Это в ряде случаев существенно сокращает общий объем памяти, несмотря на появление нового вида памяти — нанопамяти.

Сокращение количества и разрядности управляющих микрокоманд. Каждая из управляющих микрокоманд базовой ЭВМ производит проверку лишь одного из битов заданного регистра (см. параграф 4.3). Поэтому для декодирования команды или выявления индексной ячейки иногда приходится использовать длинные последовательности из таких УМК (более 20 УМК).

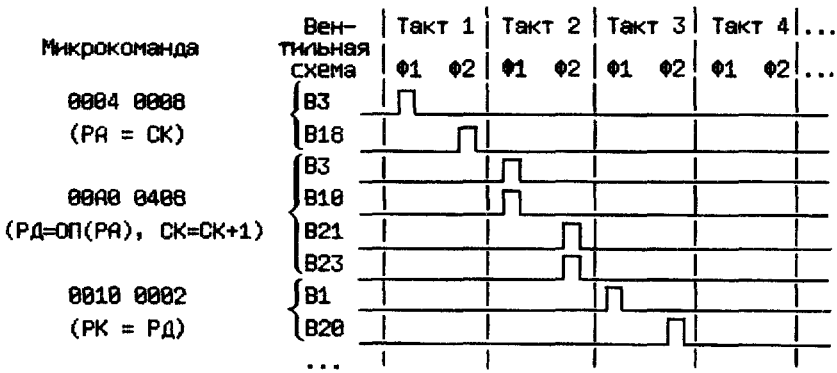


Рис. 4.12. Начало последовательности микроопераций по выборке команды

Уменьшения числа УМК можно добиться несколькими способами:

модификацией адреса, содержащегося в УМК, путем выполнения определенной логической операции над ним и каким-либо полем заданного регистра.

Это, например, позволяет использовать для декодирования кода операции команды всего лишь одну УМК вместо последовательности из 15—20 УМК;

проверкой комбинации битов заданного регистра.

Это, например, позволяет использовать для выявления индексной ячейки лишь одиу УМК вместо последовательности из 8 УМК.

Уменьшения разрядности УМК в основном добиваются путем замены адреса перехода на более короткое смещение, добавляемое к текущему адресу УМК или вычитаемое из него, и создание нескольких форматов УМК (специализацией УМК). Это позволяет получить УМК, размещаемые в однобайтовых ячейках памяти УМК, что крайне полезно в случае использования нанопамяти.

Микрокоманды различных промышленных ЭВМ чаще всего представляют собой сочетание горизонтальной и вертикальной структур с реализацией тех или иных способов сокращения микропрограммной памяти.

Глава 5

АРХИТЕКТУРА МИКРОЭВМ

5.1. Общие положения

ЭВМ — это сложная система, включающая как технические средства (интегральные микросхемы, печатные платы, кабели, источники питания, печатающие устройства, дисплеи и т. п.), так и программное обеспечение (отладчики, редакторы текста, трансляторы, загрузчики и т. д.). Для изучения ЭВМ целесообразно использовать ту или иную степень ее детализации. Здесь мы представим ЭВМ в виде трех последовательно усложняющихся уровней детализации:

1) аппаратные средства — электронные схемы, из которых строят отдельные устройства ЭВМ;

2) архитектура — состав, характеристики и взаимосвязь устройств ЭВМ (структурная организация ЭВМ), принцип функционирования ЭВМ и ее машинный язык;

3) программное обеспечение ЭВМ.

Рассмотрим с использованием этих уровней, как изменились ЭВМ за 40 лет их существования.

Первый этап (до 1955 г.). За точку отсчета эры ЭВМ принимают 1946 г., когда началась опытная эксплуатация первых образцов подобных машин. Известны такие данные о первой из них: общая масса — 30 т, число электронных ламп — 18 тыс, потребляемая мощность — 150 кВт (мощность, достаточная для небольшого завода), объем памяти — 20 10-разрядных десятичных чисел, время выполнения операций: сложения — 0,0002 с, умножения — 0,0028 с. Числа в ЭВМ вводились с помощью перфокарт и набора на переключателях, а программа задавалась соединением гнезд на специальных наборных полях. Производительность этой гигантской ЭВМ была ниже, чем

60-рублевого программируемого карманного калькулятора «Электроника МК-54» (или «Электроника Б3-34»), который можно сегодня купить в любом канцелярском магазине.

Ламповые ЭВМ имели большие габариты и массу, потребляли много энергии и были очень дорогостоящими, что резко сужало круг пользователей ЭВМ, а следовательно, объем производства этих машин. Основными их пользователями были ученые, решавшие наиболее актуальные научно-технические задачи, связанные с развитием атомной энергетики, реактивной авиации, ракетостроения и т. п. Увеличению количества решаемых задач препятствовали низкие надежность и производительность ламповых машин, ограниченность их ресурсов и чрезвычайно трудоемкий процесс подготовки, ввода и отладки программ, написанных на языке машинных команд.

Повышение быстродействия ЭВМ шло за счет увеличения ее памяти и улучшения архитектуры: использование двоичных кодов для представления чисел и команд, а также размещение их в увеличивающейся памяти ЭВМ упростили структуру процессора и повысили производительность обработки данных. Для ускорения процесса подготовки программ стали создавать первые языки автоматизации программирования (языки символического кодирования и автокоды).

Второй этап (до 1965 г.). Появление ЭВМ, построенных на транзисторах, привело к уменьшению их габаритов, массы, энергозатрат и стоимости, а также к увеличению надежности и производительности. Это сразу расширило круг пользователей и, следовательно, номенклатуру решаемых задач. Стали создавать алгоритмические языки для инженерно-технических (АЛГОЛ, ФОРТРАН) и экономических (КОБОЛ) расчетов.

Но и на этом этапе основной задачей технологии программирования оставалось обеспечение экономии машинных ресурсов (машинного времени и памяти). Для ее решения стали создавать операционные системы (комплексы служебных программ, обеспечивающих лучшее распределение ресурсов ЭВМ при исполнении пользовательских задач).

Первые операционные системы (ОС) просто автоматизировали работу оператора ЭВМ, связанную с выполнением задания пользователя: ввод в машину текста программы, вызов нужного транслятора, вызов потребовавшихся для программы библиотечных подпрограмм, вызов компоновщика для размещения этих подпрограмм и основной программы в памяти ЭВМ, ввод исходных данных и т. д. Теперь же вместе с программой и данными в ЭВМ вводилась еще и инструкция, где перечислялись этапы обработки и приводился ряд сведений о программе и ее авторе (чтобы было известно, с кого брать деньги за обработку данных). Затем в ЭВМ стали вводить сразу по несколько заданий пользователей (пакет заданий), операционные системы стали распределять ресурсы ЭВМ между этими заданиями — появился мультипрограммный режим обработки данных (например, пока выводятся результаты одной задачи, производятся расчеты для другой и в память вводятся данные для третьей).

Третий этап (до 1970 г.). Увеличения быстродействия и надежности полупроводниковых схем, а также уменьшения их габаритов, потребляемой мощности и стоимости удалось добиться за счет создания технологии производства интегральных микросхем (ИС), состоящих из десятков электронных элементов, образованных в прямоугольной пластине кремния с длиной стороны не более 1 см. Такая пластина (кристалл) размещается в небольшом пластмассовом корпусе, размеры которого, как правило, определяются только числом «ножек» (выводов от входов и выходов электронной схемы, созданной на кристалле).

Это позволило не только повысить производительность и снизить стоимость универсальных ЭВМ (больших ЭВМ), но и создать малогабаритные, простые, дешевые и надежные машины — мини-ЭВМ. Мини-ЭВМ первоначально предназначались для замены аппаратно-реализованных контроллеров (устройств управления) в контуре управления каким-либо объектом, в автоматизированных системах управления технологическими процессами, системах сбора и обработки экспериментальных данных, различных управляющих комплексах на подвижных объектах и т. д.

Появление мини-ЭВМ позволило сократить сроки разработки контроллеров. Вместо длительной процедуры разработки и создания сложной электронной схемы надо было лишь купить готовый универсальный «полуфабрикат» контроллера, чтобы потом запрограммировать его на выполнение требуемых функций. Правда, такое универсальное устройство, как правило, обладало функциональной избыточностью (для создаваемого контроллера могли не потребоваться некоторые команды мини-ЭВМ, часть ее памяти, высокое быстродействие и т. п.). Однако низкая цена серийной мини-ЭВМ, большое число серийных устройств связи с объектом управления и хорошее программное обеспечение обычно обусловливали экономическую эффективность использования такого программируемого контроллера.

Организации, покупавшие мини-ЭВМ для создания контроллеров, довольно быстро поняли, что на этих машинах можно решать и вычислительные задачи — традиционные задачи больших ЭВМ. Простота обслуживания мини-ЭВМ, их сравнительно низкая стоимость и малые габариты позволяли снабдить этими машинами небольшие коллективы исследователей, разработчиков, экспериментаторов и обучающихся, т. е. дать их прямо в руки пользователей ЭВМ. В начале 70-х годов с термином мини-ЭВМ связывали уже два существенно различных типа средств вычислительной техники:

универсальный блок обработки данных и выдачи управляющих сигналов, серийно выпускаемый для применения в различных специализированных системах контроля и управления;

небольших габаритов универсальную ЭВМ, проблемно-ориентированную пользователем на решение ограниченного круга задач в рамках одной лаборатории, технологического участка, т. е. задач, в решении которых оказывались заинтересованными 10—20 человек, работавших над одной проблемой.

Четвертый этап (до 1978 г.). Успехи в развитии электроники привели к созданию больших интегральных схем (БИС), где в одном кристалле размещалось несколько десятков тысяч электрических элементов. Это позволило разработать более дешевые ЭВМ, имеющие большую память и меньший цикл выполнения команды: стоимость байта памяти и одной машинной операции начала резко снижаться. Но так как затраты на программирование почти не сокращались, то на первый план вышла задача экономии человеческих, а не машинных ресурсов.

Разрабатывались новые операционные системы, позволяющие программистам отлаживать свои программы прямо за дисплеем ЭВМ (в диалоговом режиме), что облегчало работу пользователей ЭВМ и ускоряло разработку программ. Это полностью противоречило концепциям первых этапов информационной технологии: «процессор выполняет лишь ту часть работы по обработке данных, которую люди принципиально выполнить не могут,— массовый счет». Стала прослеживаться другая тенденция: «все, что могут делать машины, должны делать машины; люди выполняют лишь ту часть работы, которую нельзя автоматизировать».

В 1971 г. был изготовлен первый микропроцессор — БИС, в которой полностью размещался процессор ЭВМ простой архитектуры. Стала реаль-

ной возможность размещения в одной БИС (на одном кристалле) почти всех электронных устройств несложных по архитектуре ЭВМ, т. е. возможность серийного выпуска простых ЭВМ стоимостью 5—50 руб. (без учета стоимости внешних устройств). Появились дешевые микрокалькуляторы (карманные клавишные ЭВМ) и микроконтроллеры — управляющие устройства, построенные на одной или нескольких БИС, содержащих процессор, память и системы связи с датчиками и исполнительными органами в объекте управления (т. е. с внешними устройствами такой специализированной ЭВМ). Программы управления подачей топлива в двигатель автомобиля, движением электронной игрушки или заданным режимом стирки белья вводились в память ЭВМ либо при изготовлении подобного контроллера, либо непосредственно на предприятиях, выпускающих автомобили, игрушки, стиральные машины и т. п.

В 70-х годах стали изготавливать и микроЭВМ (рис. 5.1) — универсальные вычислительные системы, состоящие из процессора, памяти, схем сопряжения с устройствами ввода-вывода и тактового генератора, размещенных в одной БИС (однокристальная ЭВМ) или в нескольких БИС, установленных на одной печатной плате (одноплатная ЭВМ). Таким образом, повторялась картина 60-х годов, когда первые мини-ЭВМ отбирали часть работы у больших универсальных ЭВМ.

Пятый этап. Улучшение технологии производства БИС позволяло изготавливать дешевые электронные схемы, содержащие сотни тысяч элементов в кристалле — схемы сверхбольшой степени интеграции (СБИС).

Появилась возможность создать настольный прибор с габаритами массового телевизора, в котором размещались микроЭВМ, клавиатура, экран, кассетный магнитофон (или гибкий диск), а также схемы сопряжения с малогабаритным печатающим устройством (например, с электрической пишущей машинкой), измерительной аппаратурой, другими ЭВМ и т. д. Благодаря операционной системе, обеспечивающей простоту общения с этой ЭВМ, большой библиотеке прикладных программ по различным отраслям человеческой деятельности, а также малой стоимости такой персональный

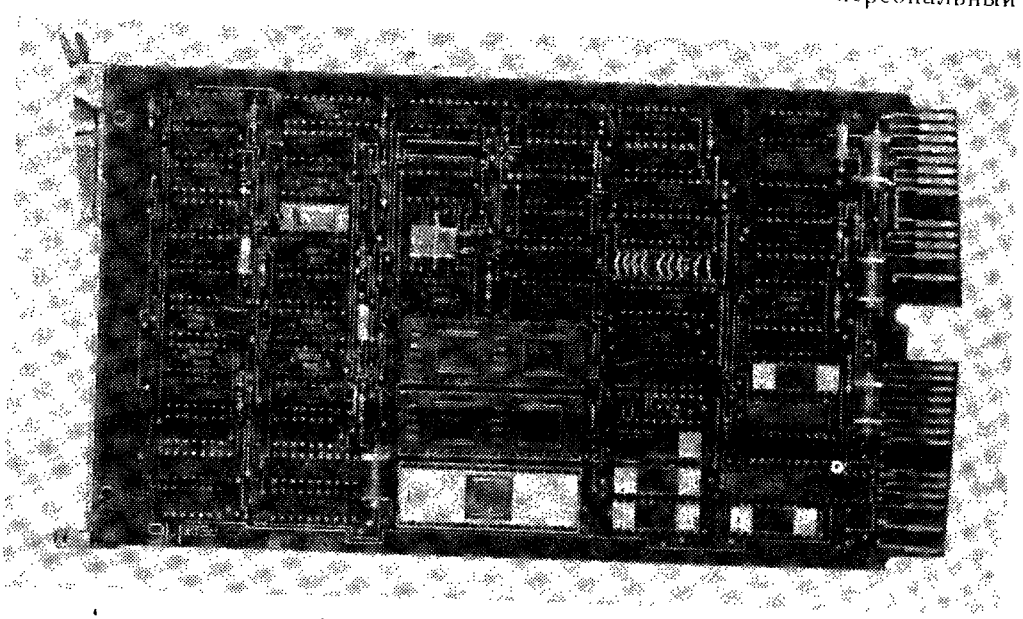


Рис. 5.1. Одноплатная микроЭВМ

компьютер становится необходимой принадлежностью инженера, исследователя, экономиста, врача, агронома, преподавателя, редактора, секретаря и даже ребенка.

Кроме функций помощника в решении традиционных задач расчетного характера персональный компьютер (ПК) может выполнять работу личного секретаря: помогать в составлении личной картотеки и ведении рабочей тетради; создавать, хранить (например, на магнитной ленте), редактировать и размножать различные тексты (письма, документы, статьи, конспекты лекций и даже книги); получать по линиям связи различные справочные данные из библиотек и архивов и т. д. ПК можно использовать для обучения детей (с помощью учебных машинных курсов), в развлекательных целях (от остросюжетных до познавательных игр) и во многих других приложениях.

Наиболее широкое применение нашли микроЭВМ в гибких системах автоматизации производства (ГАП) и научных исследований (АСНИ). Здесь используются как персональные компьютеры, так и встраиваемые в аппаратуру микроконтроллеры и микроЭВМ. Существует и множество других приложений микроЭВМ.

Для построения интеллектуальных систем и устройств выпускается очень большое число вычислительных средств: общий объем мирового производства оценивался в 1985 г. на уровне 200 млн микропроцессоров и 10 млн персональных компьютеров в год. Их встраивание в аппаратуру, обслуживание и программирование требует колоссального числа специалистов, которых нельзя (да и нецелесообразно) готовить в рамках одной специальности. Поэтому программирование постепенно перекладывается на плечи непрофессиональных программистов — специалистов различных отраслей народного хозяйства, которые приобрели начальные навыки программирования, освоив небольшие по объему дисциплины в техникумах, вузах или на курсах повышения квалификации. Эти же непрофессионалы должны выполнять и часть работ по встраиванию микроЭВМ в разрабатываемую ими аппаратуру. В связи с этим сейчас так много людей изучает в том или ином объеме устройство микроЭВМ и их язык.

Данная книга адресована тем, кто по специфике своей настоящей или будущей деятельности должен заниматься созданием технических устройств, в которые необходимо (или целесообразно) встраивать специализированные управляющую микроЭВМ (программируемый микроконтроллер) или даже универсальную микроЭВМ. Последнее позволяет пользователю самостоятельно (программным путем) изменять и (или) добавлять те или иные режимы работы технического устройства. Так как при современном состоянии вычислительной техники не позволительно проводить такую работу, рассматривая микроЭВМ как «черный ящик», то в данной и следующих главах изложены основные принципы построения и функционирования различных микроЭВМ (их архитектура). Понимание этих принципов позволит произвести правильный выбор типа микроЭВМ, объема ее памяти, набора внешних устройств и т. п.

5.2. Структуры типичных микроЭВМ

Процессоры современных микроЭВМ (микропроцессоры) обычно изготавливают в виде одной большой интегральной схемы (БИС), имеющей до 300 тыс. элементов в кристалле, или в виде одной печатной платы, на которой кроме процессора часто размещают еще генератор тактовой частоты, часть памяти ЭВМ и ряд других устройств. Такой микропроцессор должен соединяться системой шин с памятью и контроллерами внешних устройств, и

эти соединения должны осуществляться через выводы БИС или контакты разъема (разъемов) печатной платы. Определим, сколько выводов требуется процессору базовой ЭВМ, чтобы он мог обмениваться информацией с памятью и ВУ.

Для связи с памятью (см. рис. 3.1) процессору базовой ЭВМ необходимо передавать адрес (11-разрядная шина адреса), получать команды или операнды (16-разрядная шина «Чтение») или передавать результаты (16-разрядная шина «Запись»). Кроме того, необходимы провода для передачи ряда управляющих сигналов, т. е. всего около 45 проводов. Для связи с контроллерами ВУ процессор должен иметь около 30 проводов (8 — шина ввода, 8 — шина вывода, 8 — адрес ВУ, состояние флагов ВУ, запрос на прерывание и др.).

Следовательно, при изготовлении в виде БИС процессору базовой ЭВМ требуются более 75 выводов (надо подавать еще и питающие напряжения), а при изготовлении на печатной плате — два разъема по 40 или 48 контактов в каждом. Такое количество выводов БИС или печатной платы неоправдано увеличивает их геометрические размеры, массу и стоимость (в настоящее время приходится учитывать стоимость печатных соединений, разъемов и кабелей, служащих для связи между устройствами ЭВМ).

Сокращения выводов БИС (контактов разъемов) добиваются за счет использования двунаправленных шин. Так, вместо шин «Чтение» и «Запись» можно иметь лишь одну шину, по которой в зависимости от значения дополнительного управляющего сигнала (READ/WRITE) данные передаются либо из памяти в процессор, либо в обратном направлении (сокращение на 15 выводов). Аналогичную двунаправленную шину можно использовать и для связи с контроллерами ВУ (ввод или вывод по сигналу IN/OUT), благодаря чему достигается сокращение еще на 7 выводов. Полученную таким образом структуру (рис. 5.2, *а*) будем называть структурой с [illegible].

Сходство процессов обмена информацией процессора — ячеек памяти и процессора — регистров контроллеров ВУ приводит к мысли об использовании одних и тех же выводов для связи с памятью и контроллерами ВУ, т. е. об организации структуры, изображенной на рис. 5.2, *б*. При такой структуре ЭВМ (структуре с [illegible]) число выводов БИС сокращается еще на 16, но исчезает возможность одновременного обмена информацией между процессором, памятью и каким-либо ВУ. Чтобы занять шины для обмена с памятью, процессор подает управляющие сигналы *R/W* (READ/WRITE), а для обмена с ВУ — сигналы *I/O* (IN/OUT).

Дальнейшего сокращения выводов БИС микропроцессора с трехшинной структурой (шина адреса, данных и управления) можно добиться за счет объединения адресной шины и шины данных (двухшинная структура на рис. 5.2, *в*). При этом возникает так называемая [illegible] (от лат. multiplex — многократный), по которой в одни моменты времени передаются адреса, а в другие — данные. Такая структура позволила бы создать БИС микропроцессора базовой ЭВМ всего лишь с 30 выводами.

Еще большего упрощения структуры процессора и шин добиваются за счет исключения команд ввода-вывода из системы команд ЭВМ. При этом часть адресного пространства памяти (пространства, определяемого разрядностью регистра адреса) отводят для адресов регистров контроллеров ВУ и используют для ввода и вывода любые команды ЭВМ, которые осуществляют запись данных в эту «область памяти» (в базовой ЭВМ команда MOV) и чтение из нее (например, ADD, AND из состава команд базовой ЭВМ).

Структуры с вводом-выводом, организованным по аналогии с обращением к памяти (структуры с [illegible]), появились в начале 70-х годов.

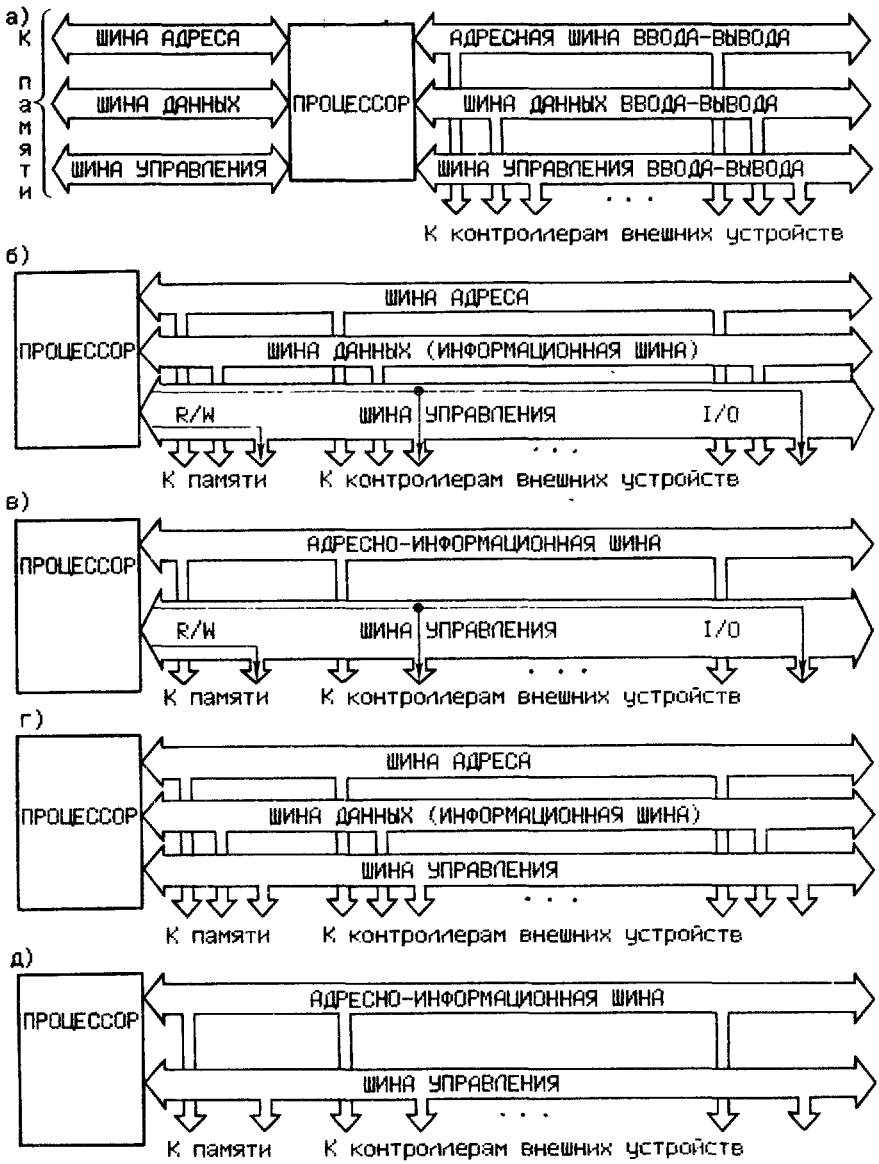


Рис. 5.2. Структуры микроЭВМ:

а — с раздельными шинами, *б* — с изолированными шинами; *в* — с изолированными шинами и мультиплексированием шин адресов и данных; *г* — с общими шинами; *д* — с общими шинами и мультиплексированием шин адресов и данных

В первой из них (структура типа представленной на рис. 5.2, *г*) при 16-разрядной адресной шине адреса с 0000 по DFFF (56K * байт) отведены для ячеек памяти, а адреса с E000 до FFFF (8K байт) — для регистров данных контроллеров ВУ. Для ввода-вывода используют команды обращения к па-

* К — символ, используемый в системах обработки данных как мера емкости памяти или скорости передачи информации, равная 1024 (2^{10}) единицам. Например, емкость памяти 64 К означает, что в ней содержатся $64 \cdot 1024 = 65\,536$ ячеек.

мяти с адресами от E000 до FFFF. Процессор как бы обманывают, заставляя его выполнять операцию с регистром данных контроллера ВУ, тогда как процессор работает так, как если бы он имел дело с ячейкой памяти ЭВМ.

Во всех современных микропроцессорах (даже в тех, которые имеют команды ввода-вывода) можно организовать ввод-вывод по аналогии с обращением к памяти. Некоторые из них обладают богатым набором команд (в том числе арифметических и логических) и способов адресации операндов, расположенных в памяти ЭВМ. Для таких микропроцессоров организация ввода-вывода по аналогии с обращением к памяти особенно выгодна.

Описание связей между процессором, памятью и контроллерами ВУ, выполненное по типу рис. 5.2, не позволяет ответить на множество чрезвычайно важных вопросов.

Каков способ передачи данных (словами, байтами или битами)?

Каков способ передачи адреса?

Какими управляющими и (или) осведомительными сигналами сопровождается передача адреса и данных?

Какие осведомительные сигналы (о состоянии памяти и ВУ) пересылаются в процессор?

Как должны быть распределены во времени информационные, управляющие и осведомительные сигналы?

Какими напряжениями кодируются двоичные 0 и 1?

С каких выводов БИС или разъема можно снять тот или иной сигнал и на какие подать?

Какой тип разъема надо использовать для подключения к процессору?

Ответы на эти и многие другие вопросы можно получить, ознакомившись с системным интерфейсом (англ. interface — общая граница).

Системный интерфейс — это набор цепей, связывающих процессор с памятью и контроллерами ВУ, алгоритм передачи сигналов по этим цепям, их электрические параметры и тип соединительных элементов.

К процессору микроЭВМ обычно подключается достаточно много внешних устройств. Это могут быть: различные датчики и исполнительные органы систем управления каким-либо объектом или прибором; устройства для долговременного хранения результатов измерения или обработки экспериментальных данных (магнитные ленты и диски); различные клавиатуры, индикаторы и печатающие устройства. Многие из этих ВУ начала выпускать промышленность задолго до появления современных микроЭВМ и иногда совсем для других целей (например, ВУ телетайп — телеграфный аппарат, имеющий клавиатуру, считыватель с перфоленты, перфоратор и печатающее устройство). ВУ свойственны различное быстродействие, различный набор управляющих сигналов, различные электрические параметры, т. е. их интерфейс (интерфейс ВУ), как правило, не совместим с системным интерфейсом микроЭВМ. Для сопряжения микроЭВМ с ВУ (системного интерфейса с интерфейсом ВУ) используют контроллеры ВУ.

Упрощения и унификации аппаратуры сопряжения добиваются за счет введения промежуточного стандартного интерфейса параллельной или последовательной передачи данных (см. параграфы 8.2, 8.3). В этом случае для подключения к микроЭВМ различных ВУ с такими стандартными интерфейсами можно использовать одинаковые (для данной ЭВМ) контроллеры параллельного или последовательного интерфейса. Такие контроллеры (иногда их называют интерфейсами или адаптерами) серийно выпускают в виде программируемых БИС для всех распространенных микроЭВМ.

Так как у терминов интерфейс, контроллер и адаптер нет общепринятых четких определений, то в литературе они часто считаются синонимами. В гл. 8 можно ознакомиться с их трактовкой в данной книге.

5.3. Организация памяти

Характеристики и принципы построения различных устройств, используемых для хранения программ и данных в микроЭВМ, рассмотрены в гл. 6 (основная память) и параграфе 7.2 (внешние запоминающие устройства). Здесь же затронуты вопросы, связанные с методами размещения и поиска информации в основной памяти микроЭВМ. По этому признаку различают адресную, стековую и ассоциативную память.

Адресная память. В простейшей ЭВМ (см. параграф 1.4), базовой ЭВМ (см. параграф 2.1) и ее микропрограммном устройстве управления (см. гл. 4) слова информации (данные, команды и микрокоманды) размещались в ячейках памяти разной длины и отыскивались по адресу (номеру) этих ячеек. Подобный способ размещения и поиска информации чаще всего использовали и используют при организации основной памяти ЭВМ. Но несмотря на то что разные микроЭВМ могут обрабатывать слова разной длины, основная их память обычно имеет байтовую организацию.

Для микроЭВМ, работающей с 8-разрядными словами, байтовая организация основной памяти является естественной. Если регистр адреса и счетчик команд такой микроЭВМ имеют, например, 16 разрядов, то она может обладать памятью из 2^{16} байт. Каждый из байтов такой памяти однозначно определяется с помощью 16-разрядного адреса (рис. 5.3, *а*). Разряды в слове (байте) нумеруются справа налево начиная с нуля.

Если микроЭВМ работает с 16-разрядными словами и имеет 16-разрядную шину данных, то при обмене данными с памятью по этой шине пересылаются 2-байтовые слова. При этом байт с младшим адресом может располагаться в слове либо справа (рис. 5.3, *б*), либо слева (рис. 5.3, *в*).

В микроЭВМ «Электроника-60» память организована по типу изображенной на рис. 5.3, *б*. Каждое 16-разрядное слово этой памяти состоит из двух последовательно расположенных байтов, первый из которых имеет четный адрес (граница слова). Например, байты 2 и 3 образуют одно полное слово, а байты 3 и 4 — нет. В командах, предназначенных для выполнения операций над словами, должны задаваться четные адреса. Если же операции производятся над байтами, то адреса могут быть любыми. При работе с такой памятью счетчик команд микропроцессора после выборки каждой команды должен наращиваться на 2, а не на 1, как в базовой ЭВМ.

В некоторых зарубежных микроЭВМ память организована по типу показанной на рис. 5.3, *в*. Такое размещение позволяет легче разрешить ряд практических проблем по хранению символов, закодированных в последова-

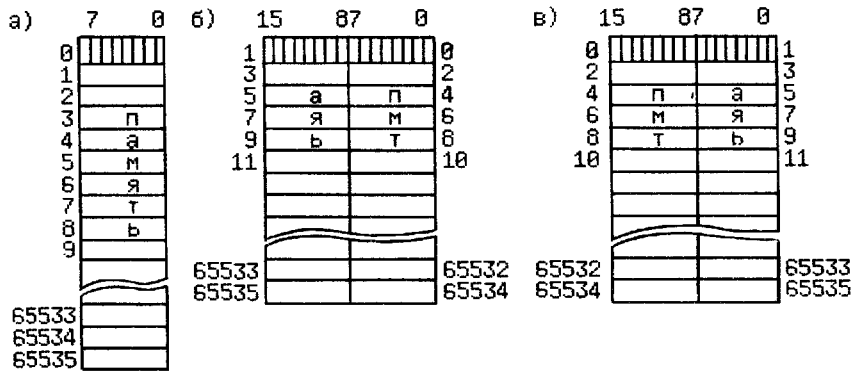


Рис. 5.3. Организация памяти:

а — для 8-разрядного процессора; *б* — для 16-разрядного процессора; *в* — для 16-разрядного процессора при другой нумерации байтов

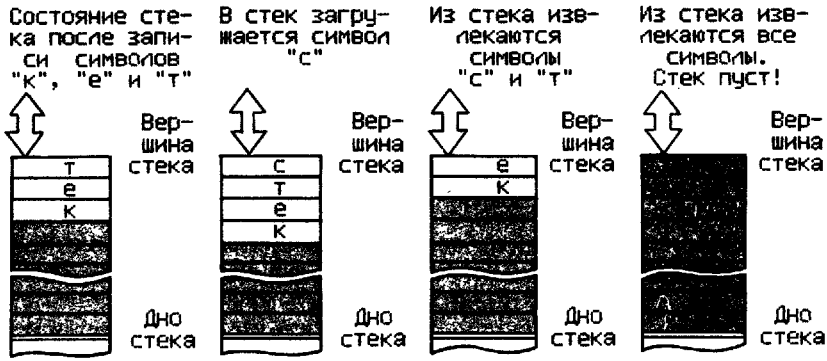


Рис. 5.4. Работа стека

тельных байтах (облегчить работу с текстами: заголовки таблиц, названия исходных данных и результатов и т. п.).

Стековая память. Она состоит из ячеек, связанных друг с другом разрядными цепями передачи слов. Обмен информацией между микроЭВМ и стековой памятью (стеком) всегда выполняется только через верхнюю ячейку — [illegible] (рис. 5.4). При записи нового слова (команды, числа, символа и т. п.) все ранее записанные слова сдвигаются на одну ячейку вниз, а новое слово помещается на вершину стека. Считывание возможно только с вершины стека и производится с удалением (после считывания все слова сдвигаются на одну ячейку вверх) или без удаления считываемого слова. Такую память часто называют памятью типа LIFO (**L**ast — **I**n **F**irst — **O**ut — последним вошел, первым вышел).

Стековая память очень удобна для упрощения решения многих задач, возникающих при работе с подпрограммами, обслуживании прерываний, построении трансляторов и т. д. Однако аппаратная реализация стека не всегда целесообразна, и поэтому в большинстве микроЭВМ стек моделируют. При этом в качестве стека обычно используют просто часть адресной памяти, что позволяет менять емкость стека и экономит аппаратуру.

Несмотря на то что внешне модель стека (рис. 5.5) отличается от аппаратного стека, она функционирует по тому же алгоритму: последним вошел, первым вышел. Это обеспечивается с помощью специального регистра процессора — [illegible], который содержит адрес плавающей

Рис. 5.5. Работа модели стека, организованного в памяти ЭВМ

вершины стека. В процессе выполнения команд, использующих стек, содержимое SP автоматически увеличивается или уменьшается на 1 (при 1-байтовых словах памяти), на 2 (при 2-байтовых словах памяти) и т. п. Так, в микроЭВМ, работающей с 1-байтовыми словами, обмен данными между процессором и стеком происходит следующим образом:

при загрузке нового слова сначала производится уменьшение содержимого SP на 1, а лишь затем запись этого слова в ячейку, на которую указывает модифицированный SP;

при извлечении слова из стека сначала выполняется чтение содержимого ячейки, на которую указывает SP, а лишь затем увеличение SP на 1.

Следовательно, то обстоятельство, что моделируемый стек растет не сверху вниз (к дну стека), а снизу вверх (от дна стека), не нарушает алгоритма функционирования такой памяти. На рис. 5.5 показан стек, организованный в памяти с 1-байтовыми словами. Дно стека расположено в ячейке 1000, так как до начала работы микроЭВМ в SP был записан адрес 1001 (стек пуст). При загрузке в стек первого слова содержимое SP уменьшится на 1 и это слово попадет в ячейку 1001 — 1 = 1000, т. е. на дно стека.

Ассоциативная память. Она обеспечивает возможность выбора информации по ее содержимому (по ассоциативному признаку). Поясним это на примере.

Пусть память ЭВМ состоит из достаточно длинных ячеек, в которых хранится информация об автомобилях, стоящих на учете в ГАИ Ленинграда:

Марка	Цвет	Номерной знак	Год выпуска	Владелец	Адрес владельца			
					Район	Улица	Дом	Квартира

Предположим теперь, что необходимо срочно получить сведения о всех черных автомобилях марки ЗАЗ-965, т. е. выбрать из памяти, декодировать и напечатать отдельные составные части (поля) всех ячеек памяти, первые десять разрядов которых содержат код

0101001001,

где 010100 — код автомобиля ЗАЗ-965, а 1001 — код черного цвета.

Если используется адресная память или стек, то процедура поиска и вывода нужной информации сводится к последовательному считыванию содержимого отдельных ячеек памяти, выделению с помощью маски

111111111100 ... 00

десяти его первых битов, сравнению выделенных битов с отыскиваемым кодом (0101001001) и выводу информации при положительном результате сравнения. Это очень нерациональная процедура, так как при малом числе черных «Запорожцев» среди сотен тысяч автомобилей Ленинграда (или даже при отсутствии таких «Запорожцев») ЭВМ должна будет прочитать и проанализировать сотни тысяч ячеек памяти.

При использовании ассоциативной памяти время поиска резко сокращается. Кроме обычных регистров (см. параграф 6.2) в ассоциативной памяти (АП) расположены дополнительные регистры ассоциативного признака (РП) и маски (РМ), число разрядов которых совпадает с числом разрядов ячеек АП, а также регистр совпадения (РС) с числом разрядов, равным числу ячеек АП. Каждый из разрядов РП и РМ связан с соответствующими разрядами ячеек АП, и если i-й разряд РМ содержит единицу (незамаскирован), то на i-е разряды всех ячеек АП поступает содержимое i-го разряда РП.

Запоминающие элементы АП снабжены логическими схемами, позволяющими сравнивать сохраняемое значение со значением соответствующего незамаскированного разряда РП. При полном совпадении информации, сохраняемой в незамаскированных разрядах РП и j-й ячейке памяти, логические схемы этой ячейки выдадут сигнал, который установит 1 в j-м разряде РС. А так как процесс сравнения информации проходит одновременно во всех ячейках АП, то за время одного обращения к такой памяти будут выявлены (и отмечены единицами в РС) все ячейки памяти, содержащие нужную информацию. Содержимое регистра совпадения поступает на формирователь результата, который выдает два сигнала, соответствующие наличию или отсутствию искомой информации и наличию одной или нескольких ячеек, содержащих эту информацию. При наличии искомой информации можно приступить к выводу содержимого лишь тех ячеек, которые отмечены единицами в РС.

Для получения информации о черных ЗАЗ-965 в ассоциативную память посылаются слова

111111111100 ... 00 и 010100100100 ... 00,

размещаемые в регистрах маски и ассоциативного признака соответственно. Затем считывается признак наличия (отсутствия) в памяти искомой информации, и если он равен единице (данные о черных «Запорожцах» есть), то производятся чтение и вывод на печать содержимого всех ячеек, которые отмечены единицей в регистре совпадения.

Некоторое усложнение запоминающих элементов и логических схем ассоциативной памяти (что вполне осуществимо при современной технологии изготовления БИС) позволяет выполнять поиск слов или чисел, расположенных в определенных границах, поиск однокоренных слов (при переводе или других манипуляциях с текстом) и т. п.

5.4. Структуры процессоров

Структуры процессоров современных микроЭВМ достаточно разнообразны. Однако большинство из них можно условно отнести к трем типам: с аккумуляторами, регистрами общего назначения и ориентированные на использование памяти со стековой организацией.

Простейшие процессоры содержат минимальное число регистров: счетчик команд, регистр команд, регистр адреса, регистр состояния, аккумулятор и буферные регистры (в базовой ЭВМ — два буферных регистра, один из которых называется регистром данных). Такие процессоры обычно называют процессорами с аккумулятором, так как все арифметические и логические операции, а также операции по передаче данных выполняются в них с использованием аккумулятора.

Для повышения производительности процессора увеличивают количество его регистров: вводят индексные регистры, указатель стека, дополнительные аккумуляторы. Это расширяет способы адресации операндов, позволяет сохранять в процессоре и использовать без обращения к памяти промежуточные результаты, работать с двухадресными командами (уменьшить число обращений к памяти), упростить работу с подпрограммами и т. д. Но пока каждый из этих дополнительных регистров процессора специализирован на выполнение лишь одной функции и во всех арифметических (логических) операциях используются не более двух аккумуляторов, такой процессор относят к процессорам с аккумуляторами.

Существуют процессоры, большинство регистров которых универсальны, т. е. могут выполнять функции аккумуляторов, индексных регистров, ука-

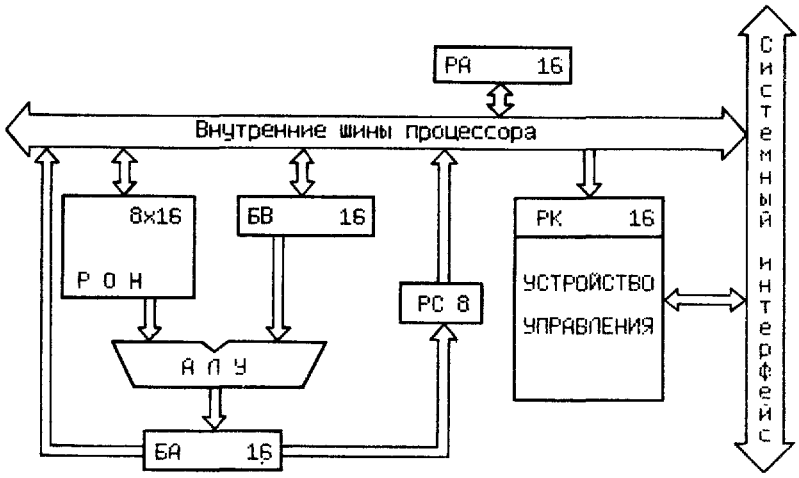


Рис. 5.6. Организация процессора с регистрами общего назначения

зателей стека и т. п. Такие регистры называют регистрами общего назначения (РОНами), а содержащие их процессоры — процессорами с регистрами общего назначения. На рис. 5.6 приведена структура гипотетического процессора с РОНами (Э60), похожего на процессор микроЭВМ «Электроника-60».

Процессор Э60 содержит восемь 16-разрядных РОНов (R0—R7), 8-разрядный регистр состояний (РС), 16-разрядные регистры адреса (РА) и команд (РК), а также два буферных регистра (БА и БВ). Арифметико-логическое устройство (АЛУ) производит операции над двумя 16-разрядными или 8-разрядными величинами с целью получения 16-разрядного (8-разрядного) результата. Устройство управления дешифрирует команды и управляет различными узлами в целях осуществления выборки и выполнения команд. Связь с памятью и внешними устройствами осуществляется через системный интерфейс (см. параграф 5.2). Отсутствие в перечне счетчика команд связано с тем, что его функции выполняет один из РОНов (R7).

Несмотря на то что для работы Э60 важны все рассмотренные выше узлы, команды непосредственно оперируют лишь с РОНами и РС, содержимое которых имеет существенное значение после выполнения каждой команды. Только эти регистры доступны программисту. Программно-доступные регистры и команды Э60 совпадают с аналогичными регистрами и командами микроЭВМ «Электроника-60» (см. приложение В). Поэтому приведенные ниже примеры и выводы в равной степени относятся как к Э60, так и к микроЭВМ «Электроника-60».

Рассмотрим несколько примеров организации циклических программ на языке микроЭВМ «Электроника-60», сравнив их с аналогичными программами базовой ЭВМ (ЭВМ с аккумулятором).

Пример 5.1. Получить произведение $Z = 50\ Y$ (сравните с табл. 2.10 и 2.11).

Если разместить 50 в регистре R1, а Y — в регистре R2, то после выполнения всего одной команды

MUL R2,R1

в R2 и R3 будет получен искомый результат умножения (младшие разряды в R3).

Для выполнения этого куска программы (этой команды) требуется всего одно обращение к памяти (выборка самой команды), в соответствующем же куске программы базовой ЭВМ — 300 обращений к памяти: цикл, в котором из памяти выбираются команды ADD, ISZ, BR и их

операнды, повторяются 50 раз. И хотя команда умножения выполняется в 20—25 раз дольше, чем команда сложения и (или) перехода, общее время выполнения рассматриваемой программы на ЭВМ с РОНами много меньше, чем на ЭВМ с аккумулятором.

Пример 5.2. Получить сумму элементов массива (сравните с табл. 2.12—2.14).

Если разместить в R1 количество элементов массива, в R2 — адрес первого его элемента и отвести R3 для записи результата (суммы элементов), то программа решения поставленной задачи может иметь следующий вид:

```
   CLR R3          ;очистка регистра результата
A:ADD (R2)+,R3     ;суммирование очередного элемента
   SOB R1, A       ;уменьшение числа элементов на 1 и переход к метке A.
                   ;если это число не равно 0
   HALT            ;останов
```

Цикл, выполняемый в этой программе, требует всего трех обращений к памяти (выборка команд ADD, SOB и операнда по адресу, указанному в R2), а аналогичный цикл программы в табл. 2.14 — семи обращений (выборка команд ADD, ISZ, BR, адреса, на который указывает ячейка F, и трех операндов).

Примеры 5.1 и 5.2 показывают, что программы, составленные для ЭВМ с РОНами, могут реже обращаться к памяти, чем программы, составленные для ЭВМ с аккумулятором. А так как время, необходимое для чтения команд (операндов) из памяти ЭВМ или записи результата в память (см. гл. 6), значительно больше, чем время, требуемое для чтения операнда из регистра процессора или записи в него результата вычислений, то ЭВМ с РОНами эффективнее ЭВМ с аккумулятором.

У процессоров со стековой организацией нет ни аккумуляторов, ни РОНов. В простейшем случае их программно-доступные регистры (рис. 5.7, *а*) служат для указания адресов вершины стека (указатель стека) и следующей команды программы (счетчик команд), а также хранения состояний отдельных устройств процессора (регистр состояний). Память ЭВМ со стековой организацией разделяется на две части (рис. 5.7, *б*), одна из которых (например, нижняя) отводится под моделируемый стек (см. параграф 5.3), а другая — под программу и «глобальные» данные.

Подобная организация позволяет строить программы, в которых используются:

адресные команды для загрузки операнда из области данных в стек (PUSH) или извлечения операнда из стека (POP);

безадресные команды, инициирующие извлечение из стека двух слов (или одного слова), выполнение над ними действий, указанных кодом операции команды, и загрузку результата в стек.

При составлении программы вычисления арифметических выражений для таких ЭВМ используют польскую инверсную (бескобочную) запись

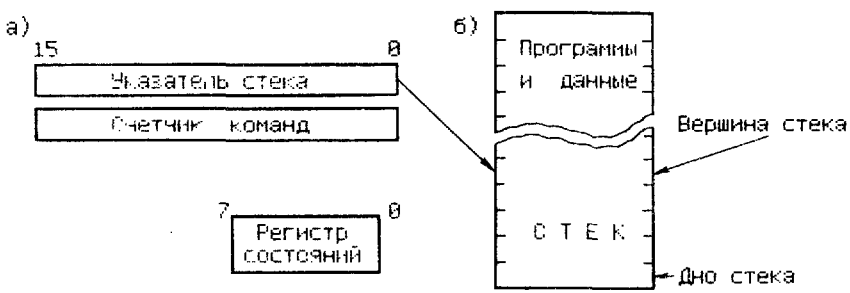


Рис. 5.7. Регистры и память ЭВМ со стековой организацией

Таблица 5.1

Вычисления по формуле $8+2\times5/(6-4)$ путем использования бесскобочной записи этой формулы: $825\times+64-/$

Шаг	Формула, подлежащая расчету	Левый знак операции	Операнды	Результат	Новая формула после выполнения операции
1	8 2 5 × + 6 4 − /	×	2 и 5	10	8 10 + 6 4 − /
2	8 10 + 6 4 − /	+	8 и 10	18	18 6 4 − /
3	18 6 4 − /	−	6 и 4	2	18 2/
4	18 2/	/	18 и 2	9	9

выражений. В бесскобочной записи операция записывается не между операндами (X+Y), а после них (XY+), т. е. выражение

$$3\ln(X+1)-A/X^2$$

переписывают в виде

$$3X1+\ln\times AX2\uparrow/-,$$

где × и ↑ используют для указания операций умножения и возведения в степень соответственно.

Подобные выражения можно вычислять (на бумаге) по следующему алгоритму:

1) проанализировать каждый символ бесскобочной записи формулы (начиная с самого левого символа) до тех пор, пока не встретится знак операции;

2) взять два ближайших операнда, расположенных слева от обнаруженного знака операции, выполнить над ними заданную операцию и поместить результат обратно в формулу на место использованных операндов и знака операции;

3) если теперь формула состоит из одного значения, то это значение и есть результат, т. е. алгоритм завершен; в противном случае необходимо перейти к выполнению шага 1.

Пример использования такого алгоритма для вычисления по формуле

$$Y=(A+B\times C)/(E-F),$$

преобразованной в форму

$$ABC\times+EF-/, \qquad (5.1)$$

приведен в табл. 5.1. Для определенности в ней были выбраны следующие числовые значения операндов: A=8, B=2, C=5, E=6 и F=4.

Бесскобочная запись является идеальной для проведения вычислений на ЭВМ со стековой организацией. На рис. 5.8 изображен расчет выражения (5.1) с использованием стека. В расчете используются адресные команды *PUSH* и *POP*, в которых адреса операндов *A*, *B*, ... указываются как *ад. A*,

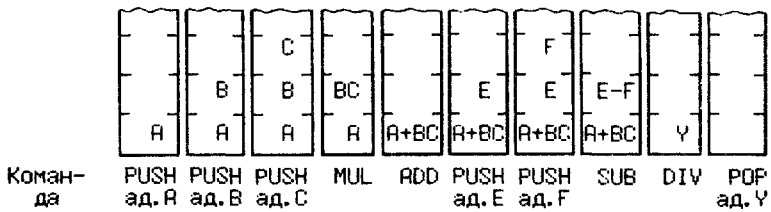


Рис. 5.8. Содержимое стека после выполнения операций по реализации выражения (5.1)

ад. B, ... Кроме того, в нем используются безадресные команды умножения (*MUL*), сложения (*ADD*), вычитания (*SUB*) и деления (*DIV*), выполняющие операции с операндом, находящимся на вершине стека, и операндом, расположенным непосредственно под ним.

Машины со стековой организацией имеют большие преимущества при выполнении сложных вычислений, их команды короче и занимают меньше места в памяти ЭВМ. Многие настольные и карманные калькуляторы представляют собой ЭВМ со стековой организацией. Однако быстродействие ЭВМ с рассматриваемой организацией, как правило, ниже, чем быстродействие ЭВМ с аккумуляторами или РОНами, поскольку единственным «рабочим регистром», используемым при выполнении арифметических операций, является вершина стека. А так как стек находится в памяти ЭВМ, обмен с которой сравнительно долог, то это «медленный» регистр.

Структуры реальных процессоров микроЭВМ не всегда можно с полной определенностью отнести к разряду машин с аккумуляторами, регистрами общего назначения или со стековой организацией. Так, типичная микроЭВМ с РОНами — «Электроника-60» позволяет моделировать все рассмотренные структуры. Например, при моделировании машин со стековой организацией адресные команды загрузки $PUSH_{ад.A}$ и извлечения $POP_{ад.B}$ имитируются в ней командами

```
MOV @#ад.A,-(R6)  и  MOV (R6),@#ад.B ,
```

а безадресная команда ADD — командой ADD (R6)+, (R6). Микропроцессор КР1810ВМ86 (см. параграф 10.3) при выполнении одних операций (например, умножения и деления) может быть отнесен к процессорам с аккумулятором, а при выполнении других операций — к процессорам с РОНами. Однако классификация способствует пониманию архитектур различных процессоров и, следовательно, оценке эффективности их работы.

Эффективность работы процессора и объем памяти зависят не только от структуры процессора, но и от набора его команд, способов адресации операндов, числа адресов, задаваемых в типичной команде, разрядности операндов и ряда других параметров. Примером тому могут служить фрагменты программ суммирования двух 16-разрядных чисел, расположенных в ячейках памяти с адресами X и Y, с пересылкой результата в ячейку памяти с адресом Y: фрагмент для 16-разрядной микроЭВМ с РОНами «Электроника-60»

```
ADD @#X,@#Y ;сложить содержимое ячеек с адресами X и Y,
            ;а результат переслать в ячейку с адресом Y
```

фрагмент для 16-разрядной базовой ЭВМ (ЭВМ с аккумулятором)

```
CLA   ;очистить аккумулятор
ADD X ;сложить  содержимое ячейки с адресом X и аккумулятора
ADD Y ;сложить  содержимое ячейки с адресом Y и аккумулятора
MOV Y ;переслать  сумму  в ячейку с адресом Y
```

фрагмент для 8-разрядного микропроцессора КР580ИК80А (ЭВМ с аккумулятором)

```
LXI B,X ;загрузить адрес X в пару регистров B и C
LXI H,Y ;загрузить адрес Y в пару регистров H и L
LDAX B  ;загрузить аккумулятор байтом с адресом,
        ;записанным в паре регистров B и C
```

```
ADD M    ;сложить  содержимое аккумулятора и байта с адресом,
         ;записанным в паре регистров H и L
STAX H   ;переслать содержимое аккумулятора по адресу,
         ;записанному в паре регистров H и L
INX B    ;вычислить адрес старшего байта первого слагаемого,
         ;увеличив на 1 содержимое пары регистров B и C
INX H    ;вычислить адрес старшего байта второго слагаемого,
         ;увеличив на 1 содержимое пары регистров H и L
LDAX B   ;загрузить аккумулятор байтом с адресом,
         ;записанным в паре регистров B и C
ADC M    ;сложить содержимое аккумулятора,  бита переноса и
         ;байта с адресом, записанным в паре регистров H и L
STAX H   ;переслать содержимое аккумулятора по адресу,
         ; записанному в паре регистров H и L
```

5.5. Немного о конструкции микроЭВМ

Кажется естественным, что увеличение числа элементов (деталей), из которых собирается какое-либо устройство, должно привести к росту его габаритов, энергопотребления и стоимости, а также к снижению надежности работы. Но все это как бы не имеет отношения к изделиям современной микроэлектроники (радиоприемникам, телевизорам, калькуляторам, персональным ЭВМ и т. д.), которые выполняют больше более сложных функций, чем их предшественники, при меньших размерах, массе, потребляемой энергии и стоимости. Как же объяснить этот феномен?

Печатные платы. Еще в «эпоху» широкого использования радиоламп начали разрабатывать способы ускорения и удешевления сборки схем, состоящих из электрических элементов (ламп, резисторов, конденсаторов). Одним из них было размещение элементов и ламповых панелек на плате с рисунком, состоящим из плоских проводников и контактных площадок с отверстиями (рис. 5.9, *а*). Рисунок создавался с помощью тонкого слоя меди, нанесенного на твердую пластину из непроводящего материала (о способах нанесения рисунка будет рассказано ниже). Элементы вставляли в отверстия контактных площадок со стороны, противоположной рисунку (рис. 5.9, *б*), а затем металлизированную сторону платы опускали на некоторое время в ванну с расплавленным припоем. Таким образом, одновременно образовывались все электрические соединения между элементами, которые жестко закреплялись на плате (припаивались к ней).

В результате исчезли различные детали, используемые ранее для закрепления элементов при монтаже с помощью проводников, отпала необходимость зачистки монтажных проводников и увязки их в жгуты, уменьшился брак монтажа, так как исключались неверные соединения элементов, правильно установленных на отведенные им места, резко уменьшилось число монтажников и т. п. Следовательно, применение печатных плат способствовало выпуску более единообразных, бездефектных и дешевых изделий, если рисунок платы был точен и хорошо воспроизводился.

Чтобы детальнее ознакомиться с дешевыми серийными печатными платами, Вы можете заглянуть в корпус своего транзисторного радиоприемника, кассетного магнитофона или другого бытового электронного прибора. Правда, там Вы скорее увидите двусторонние печатные платы, появившиеся при усложнении схем, размещаемых на печатных платах.

Усложнение схем привело к трудностям в создании рисунка с непересекающимися межэлементными соединениями. Эти трудности исчезают при

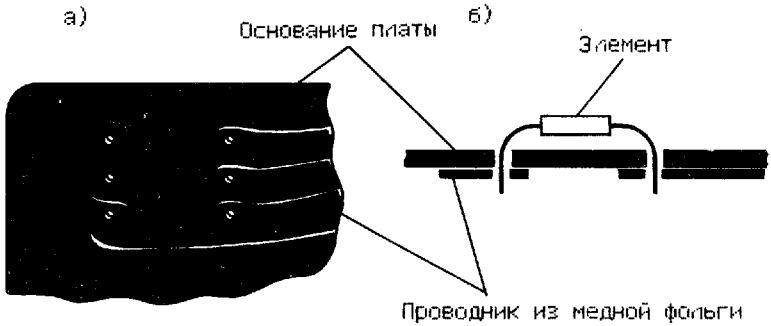


Рис. 5.9. Участок печатной платы

использовании двустороннего печатного монтажа: в том месте, где один печатный проводник должен был бы пересечься с другим, первый переводится через отверстие на противоположную сторону платы, а затем возвращается обратно. В переходные отверстия впаивают перемычки, а в более дорогих платах эти отверстия металлизируют насквозь, что избавляет изготовителя от лишних паек.

Двусторонний монтаж позволяет уменьшить размер платы за счет распределения межэлементных соединений на обе ее стороны. Для еще большего сокращения площади платы иногда используют более дорогие многослойные печатные платы (платы, склеенные из нескольких тонких одно- или двусторонних печатных плат).

Создание печатной платы начинают с вычерчивания рисунка (разводки печатной платы), в несколько раз превышающего действительные размеры схемы. Этот рисунок служит в качестве образца при изготовлении плаката: чистой пленки, на которую наклеивают готовые шаблоны контактных площадок для различных элементов и кусочки черной ленты той или иной ширины, имитирующие межэлементные соединения. Плакат фотографируют с уменьшением, чтобы получить негатив печатной платы рабочего размера. Для двусторонних или многослойных печатных плат создают аналогичные негативы остальных сторон (слоев).

Затем берут заготовку платы — кусок фольгированного медью слоистого пластика. Это может быть часть листа из пропитанных эпоксидной смолой и спрессованных слоев бумаги (гетинакс) или стекловолокна (стеклотекстолит), покрытых с одной или с обеих сторон тонкой медной фольгой (более тонкой, чем фольга от конфетной обертки). На первой стадии сверлят отверстия либо по трафарету, либо на автоматическом сверлильном станке, настроенном точно по размерам отверстий платы. Отверстия затем можно металлизировать насквозь (путем многослойного осаждения меди), благодаря чему создаются проводящие соединения с одной стороны платы на другую.

Далее плату покрывают тонкой пленкой светочувствительного лака, называемого фоторезистом, на эту пленку накладывают негатив с рисунком и производят засветку специальным световым источником (например, ультрафиолетовым). Свет полимеризует участки фоторезиста, находящиеся под изображением проводников и контактных площадок, а проявитель растворяет неполимеризованные участки пленки фоторезиста, открывая те поверхности медной фольги, которые надо убрать с платы. После этого плату обрабатывают составом, травящим медь, и на ней остается требуемый рисунок из меди, покрытой полимеризованным фоторезистом. Наконец, снимают остаток фоторезиста и все проводники (или только их контактные площадки) покрывают тонким слоем припоя.

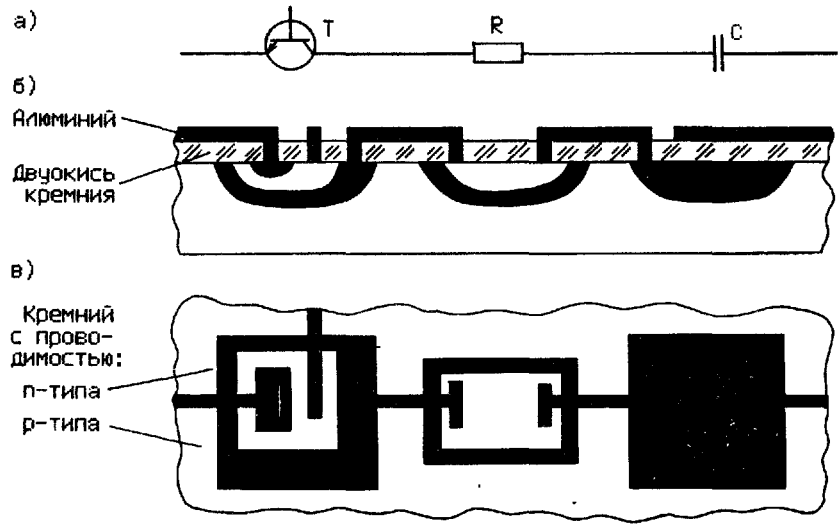


Рис. 5.10. Часть интегральной микросхемы:

а — воспроизводимая электрическая схема; *б* — поперечное сечение; *в* — вид сверху (пленка двуокиси кремния считается абсолютно прозрачной)

При изготовлении двусторонних печатных плат следует заботиться о точном совмещении рисунков на обеих сторонах платы (совмещение означает, что рисунок на каждой поверхности должен точно соответствовать отверстиям на плате и рисунку на противоположной ее стороне). То же относится к многослойным печатным платам.

Интегральные схемы. Печатные платы не позволили существенным образом уменьшить габарит ламповых схем первых ЭВМ: необходимость отвода большого количества теплоты, выделяемой этими лампами, заставляла располагать лампы на значительном расстоянии друг от друга (да и сами лампы были достаточно велики). Положение изменилось с появлением транзисторов. * Однако и они (давшие возможность резко уменьшить энергопотребление и габариты схем) не позволили существенно упростить и удешевить сборку схем, а также повысить надежность работы схем. Действительно, со сменой ламп на транзисторы почти не уменьшились число паек на печатных платах и число разъемов для связи плат между собой, но ненадежность именно этих соединений является основной причиной отказов в работе электронных схем (окисление и ряд других причин). И только с 1959 г., когда была предложена планарная технология (от лат. planum — плоскость) создания сложных схем на поверхности маленького кристалла кремния, началось последовательное уменьшение габаритов и стоимости, а также рост надежности схем микроэлектроники.

На рис. 5.10 показан сильно увеличенный кусочек кремния, на поверхности которого размещена небольшая часть сложной электронной схемы. Эта часть состоит из *n—p—n*-транзистора, резистора (его сопротивление определяется размерами и проводимостью диффузионной области *p*-типа, в которой и образован резистор), конденсатора, образованного диффузионной областью *n*-типа и пленкой алюминия, разделенных тонким слоем двуокиси кремния (диэлектрик).

---

* Читателям, желающим освежить в памяти сведения об устройстве и работе транзистора, советуем обратиться к книге Е. Айсберга «Транзистор?.. Это очень просто» (М., Энергия, 1977. 152 с.).

Указанные элементы (как и тысячи других элементов интегральной схемы) изготовлены на тонкой пластинке кремния (0,25—0,4 мм), имеющей проводимость *p*-типа. Сначала эта пластина отрезается от цилиндрического кристалла, ее верхняя сторона полируется, химически травится до зеркальной гладкости и окисляется путем нагревания в кварцевой трубке, через которую пропускается кислород или водяной пар. Полученная при этом пленка двуокиси кремния (0,3—1,3 мкм), как и аналогичная пленка, создаваемая в процессе изготовления схемы, используется в качестве изолятора между отдельными составляющими элементов, а также в качестве защитного покрытия при формировании элементов схемы и ее работе после изготовления.

Затем переходят к созданию на поверхности пластины отдельных составляющих элементов схемы, для чего удаляют в определенных местах защитный слой двуокиси кремния и внедряют в открывшийся кремний примесь, которая изменяет тип его проводимости. Последняя операция называется локальным (от лат. localis — местный) легированием (нем. legieren — связывать, сплавлять) и выполняется путем диффузии или ионного внедрения примесей.

Указанный процесс осуществляется с помощью фотолитографии в следующем порядке.

1. Двуокись кремния покрывается фоторезистом (о нем было рассказано при рассмотрении технологии изготовления печатных плат).

2. Фоторезист засвечивается через фотошаблон, выполненный так, чтобы после проявления в фоторезисте образовались окна над теми местами двуокиси кремния, которые должны быть удалены для получения доступа к поверхности кремния.

3. Фоторезист проявляется, и пластина обрабатывается слабым раствором кислоты, растворяющей все участки двуокиси кремния, над которыми нет полимеризованного светом фоторезиста; затем полимеризованный слой резиста удаляется с поверхности двуокиси кремния.

4. Над нагретой пластиной продувается обогащенный примесями азот, благодаря высокой температуре (порядка 1200 °С) атомы кремния становятся подвижными и примеси из газа легко проникают в кристаллическую решетку кремния (в области, защищенной двуокисью кремния, диффузии почти нет, поскольку окисный слой не пропускает примесей).

5. На поверхности пластины выращивается новый защитный слой двуокиси кремния, и она оказывается готовой для дальнейших изменений структуры (изготовления новых окон и т. д.).

Для реализации схемы на рис. 5.10 требуется еще дважды выполнить описанные выше операции, чтобы сначала легировать часть созданных областей примесями *p*-типа (образование базы транзистора и диффузионного резистора), а затем снова примесями *n*-типа (образование эмиттера).

Заключительные операции по изготовлению схемы связаны с организацией межэлементных соединений: созданием окон над контактными площадками, нанесением алюминиевой пленки, слоя фоторезиста, засвечиванием фоторезиста через фотошаблон с рисунком межэлементных связей, проявлением (для получения окон в фоторезисте) и, наконец, травлением открытых областей алюминия.

Следует отметить, что интегральные схемы, изготавливаемые по современной планарной технологии, чаще всего создают в эпитаксиальном («лежащем сверху») слое кремния с четко определенной физико-химической структурой. Этот слой толщиной 8—10 мкм выращивают на кремниевой или инородной (например, сапфировой) подложке. Такое усложнение технологии на порядок уменьшает брак при изготовлении микросхем.

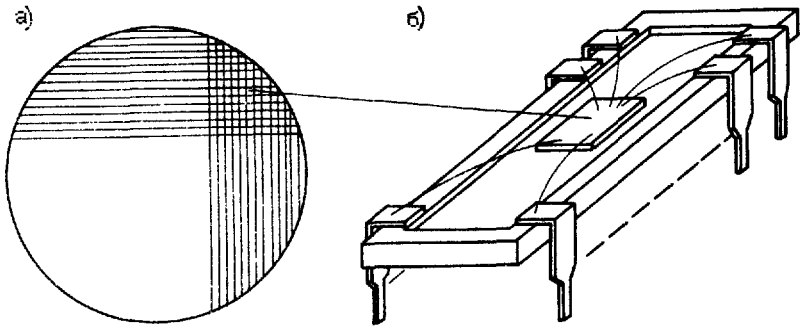
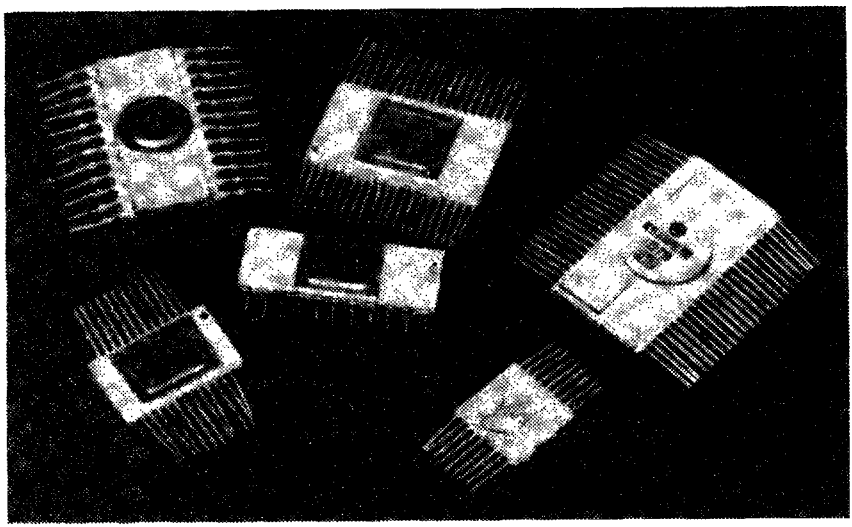


Рис. 5.11. Часть процесса изготовления интегральных схем



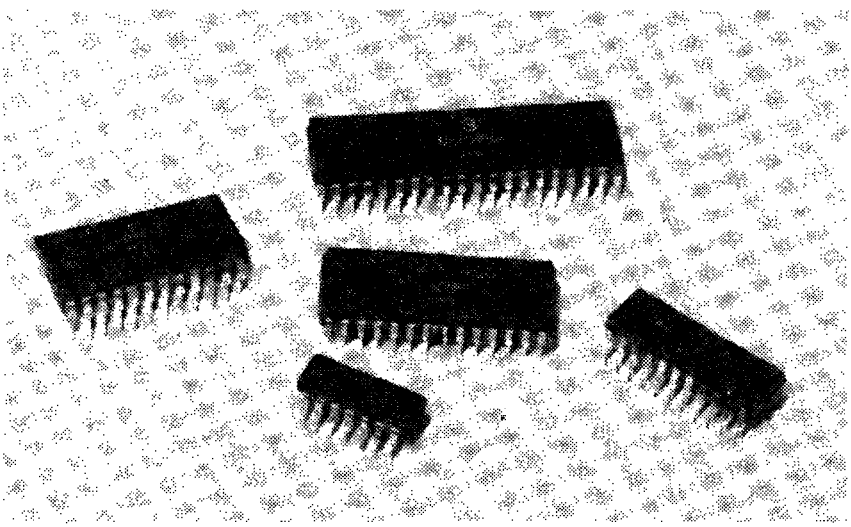


Рис. 5.12. Корпуса интегральных микросхем

Особенностью планариой технологии является ее универсальность. Основная часть технологического процесса состоит из трех повторяющихся операций (фотолитографии, термической обработки и химической обработки). Самые разнообразные интегральные схемы создают только изменением комплекта фотошаблонов и режимов термических процессов. Кроме того, по этой технологии производят групповое изготовление интегральных схем.

На одной пластинке кремния (рис. 5.11, *а*) одновременно изготавливают много однотипных схем (фотошаблон многократно проецируется на разные участки кремниевой пластины диаметром 50—120 мм). Число этих схем может достигать нескольких сотен. После изготовления элементов структур, получения металлической разводки между элементами и металлических контактных площадок для присоединения внешних выводов корпуса интегральной схемы пластина кремния разрезается на отдельные кристаллы, содержащие одну интегральную схему. Далее выполняют более трудоемкий процесс установки каждого из кристаллов в свой корпус, присоединение выводов (рис. 5.11, *б*), сборку корпуса и его герметизацию, а также комплексную проверку пригодности полученного изделия для работы в различных условиях.

Планарная технология начала 80-х годов позволяла разместить более 1000 электрических элементов на площади 1 мм^2. При размерах кристаллов от 1×1 до 10×10 мм на них размещали от 1 (например, одиночный транзистор) до 100 000 элементов. Причем стоимость таких кристаллов практически не зависела от сложности схемы. Не влияет сложность схемы и на габарит корпуса, в который помещают кристалл с этой схемой (габарит корпуса зависит от числа выводов для связи кристалла с внешней средой). Последнее легко проверить, ознакомившись со схемами БИС в гл. 10. На рис. 5.12 показаны типичные корпуса интегральных микросхем.

Улучшение методов выращивания кристаллов кремния и эпитаксиального выращивания пленок кремния, рост точности изготовления рисунков фотошаблонов и способов переноса этих рисунков на фоторезист, совершенствование химических и термических процессов и другие усовершенствования планарной технологии позволили перейти к созданию сверхбольших интегральных схем (СБИС), на которых размещаются более миллиона элементов. СБИС особенно удобны для размещения регулярных схем запоминающих устройств (см. гл. 6), на них строятся очень сложные процессорные структуры и ряд других функционально завершенных схем, имеющих малое число связей для обмена информацией с другими устройствами вычислительной системы, построенной с использованием СБИС.

Стандартная интегральная схема содержит около 100 элементов, стандартная большая интегральная схема — на два порядка больше чем ИС, т. е. около 10^6 элементов. Но несмотря на существенное различие в сложности, каждая из этих схем при испытаниях, приемке и эксплуатации рассматривается как единое целое: один элемент, состоящий из корпуса с ножками (выводами). Поэтому рост степени интеграции (при малой разнице в стоимости одного корпуса) позволяет из года в год получать все более сложные схемы ЭВМ, имеющие бóльшую надежность и меньшую стоимость.

Журнал «В мире науки» (1983, № 10) приводил интересное сравнение: «Если бы авиапромышленность в последние 25 лет развивалась столь же стремительно, как промышленность средств вычислительной техники, то сейчас самолет Boeing 767 стоил бы 500 долл. и совершал облет земного шара за 20 мин, затрачивая при этом пять галлонов топлива. Приведенные цифры весьма точно отражают снижение стоимости, рост быстродействия и повышение экономичности ЭВМ».

Глава 6

ПОДСИСТЕМА ПАМЯТИ

6.1. Характеристики запоминающих устройств

Память (или ЗУ) состоит из огромного числа элементов памяти, каждый из которых может находиться в одном из двух состояний, кодируемых двоичной цифрой 0 или 1. Элементы ЗУ (или, лучше сказать, биты, кодирующие состояния этих элементов) группируются в слова, т. е. такие порции информации, которые могут одновременно пересылаться между памятью и процессором и (или) обрабатываться последним. Поиск нужного слова памяти можно производить либо по его адресу (адресные ЗУ), либо по его содержанию (ассоциативные ЗУ).

Запоминающие устройства имеют ряд показателей качества, характеризующих их информационные и временные свойства. Основными из этих характеристик являются следующие.

Информационная емкость (или просто емкость). Емкость памяти выражается в количестве битов, байтов или слов, состоящих из определенного числа байтов. Так как эта емкость может быть очень велика (до 10^{12} бит), то обычно используют более крупные единицы, образованные присоединением приставок К (кило), М (мега) или Г (гига) к перечисленным выше единицам. При этом надо учитывать, что в системах передачи и обработки информации приставки К, М и Г соответственно равны $2^{10} = 1024$, $2^{20} = 1\,048\,676$ и $2^{30} = 1\,073\,741\,824$.

Временные соотношения. Время доступа — временной интервал, определяемый от момента, когда процессор выставил на адресной шине адрес требуемой ячейки памяти и послал по шине управления приказ на чтение или запись данных, до момента осуществления связи адресуемой ячейки с шиной данных (рис. 6.1).

Операции, выполняемые для осуществления такой связи, рассмотрены в параграфе 6.2 (см. рис. 6.5).

При операции ЧТЕНИЕ содержимое шины данных может быть переписано в соответствующий регистр процессора (RG) лишь после завершения этого интервала времени. В связи с этим память, как правило, информирует процессор о выдаче требуемых данных посылкой сигнала ГОТОВО (READY) по шине управления.

При выполнении операции ЗАПИСЬ новое содержимое адресуемой ячейки памяти должно выставляться процессором на шину данных до завершения времени доступа. Поэтому запись переданного слова может выполняться памятью непосредственно за операциями по связи шины данных с адресуемой ячейкой.

Время записи — интервал времени, необходимый для переписи содержимого шины данных в связанную с ней ячейку памяти. Минимальная величина этого интервала определяется физическими свойствами элементов памяти. При изготовлении памяти в нее включают схему, которая вырабатывает импульс записи нужной длительности. Сигнал на выработку импульса записи выдается в конце интервала доступа. После завершения импульса записи память информирует процессор о выполнении операции ЗАПИСЬ посылкой сигнала ГОТОВО по шине управления (см. рис. 6.1).

Время восстановления необходимо для приведения памяти в исходное состояние после того, как процессор снял с шин адрес, сигнал ЧТЕНИЕ или ЗАПИСЬ и данные (при выполнении операции ЗАПИСЬ). Эти сигналы сни-

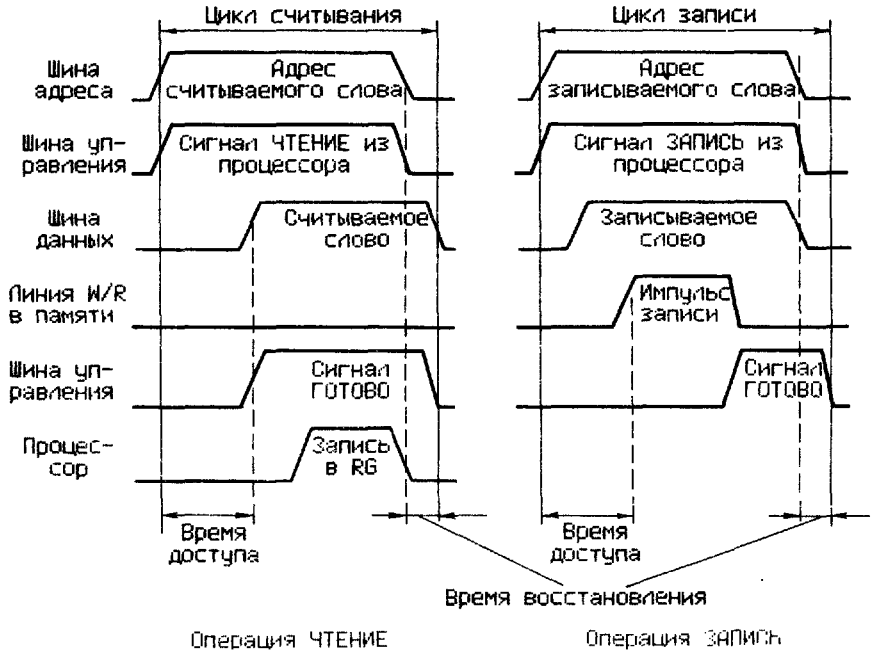


Рис. 6.1. Типичные временные диаграммы работы с памятью

маются после записи считанного слова в регистр RG (при операции ЧТЕНИЕ) или в момент получения из памяти сигнала ГОТОВО (при операции ЗАПИСЬ).

Цикл считывания и цикл записи определяются как время с момента выдачи процессором адреса требуемой ячейки памяти и сигнала на считывание или запись (они могут выдаваться позднее адреса) до того момента, когда заканчиваются все действия, связанные с выполняемой операцией, и память будет готова реализовать следующую операцию. Иногда, при совпадении продолжительности этих циклов, используют обобщающий термин — цикл памяти.

Стоимость 1 бита. Она определяется отношением стоимости памяти к ее информационной емкости.

Возможность изменения информации. Данное качество характеризует тип памяти. Существуют элементы памяти с легко изменяемыми состояниями. Память с произвольным доступом к словам, построенная на таких элементах, обычно называется оперативной памятью или оперативным запоминающим устройством (ОЗУ). Есть более дешевые элементы памяти, в которые единожды записывают 0 или 1 (при изготовлении или перед установкой в создаваемую ЭВМ). ЗУ, построенные на таких элементах, соответственно называют ПЗУ (постоянное запоминающее устройство) и ППЗУ (программируемое постоянное запоминающее устройство). Одним из вариантов ППЗУ является СППЗУ (стираемое ППЗУ). Содержимое СППЗУ можно стереть в специальной установке (для этого требуется значительное время), а затем вновь заполнить информацией.

Способ выборки информации. Различают два основных типа ЗУ: с произвольной выборкой и последовательной выборкой. В первых время доступа к заданному слову не зависит от месторасположения этого слова в памяти, а во вторых — зависит.

Сохранение информации при отключении питания ЭВМ. В некоторых видах ЗУ (например, полупроводниковых ОЗУ) происходит потеря записанной информации при отключении (и даже кратковременном прерывании) питающего напряжения. Такие устройства называют энергозависимыми ЗУ. В энергонезависимых ЗУ (например, магнитной памяти) информация при отключении питания сохраняется. Все ПЗУ и ППЗУ энергонезависимы.

Кроме приведенных характеристик довольно часто указывают и другие практически важные показатели качества ЗУ, например: потребляемую мощность, способ организации, физические свойства запоминающих элементов. В большинстве случаев требуются дешевые энергонезависимые ЗУ большой емкости с малым временем выборки, однако реализация этих требований в совокупности представляет собой в настоящее время практически неразрешимую задачу.

6.2. Конфигурация ЗУ с произвольным доступом

В настоящее время наибольшее распространение получили полупроводниковые ЗУ, построенные на БИС (СБИС), каждая из которых содержит большое число запоминающих элементов. Эти элементы обычно объединяются в M ячеек длиной 1, 4 или 8 бит. Так, БИС, содержащие 2К (2048) запоминающих элементов, можно изготавливать для хранения 2К 1-битовых слов, 512 4-битовых слов или 256 8-битовых слов (2К $\times$ 1, 512 $\times$ 4 или 256 $\times$ 8).

Существуют две основные организации ячеек запоминающих элементов внутри БИС и самих БИС в памяти ЭВМ: с одномерной адресацией и двумерной адресацией.

Кристалл ПЗУ 256$\times$8 с одномерной адресацией (рис. 6.2, *а*) содержит 256 (2^8) строк по восемь элементов в каждой. При поступлении на его входы адреса выбираемой ячейки памяти на одном из 256 выходов дешифратора появляется сигнал, этот сигнал поступает на входы «Выбор элемента» (рис. 6.2, *б*) восьми запоминающих элементов (ЗЭ), подключенных к выходу дешифратора. В зависимости от содержимого ЗЭ на его линии считывания может появиться сигнал, соответствующий 0 или 1. Таким образом, на верти-

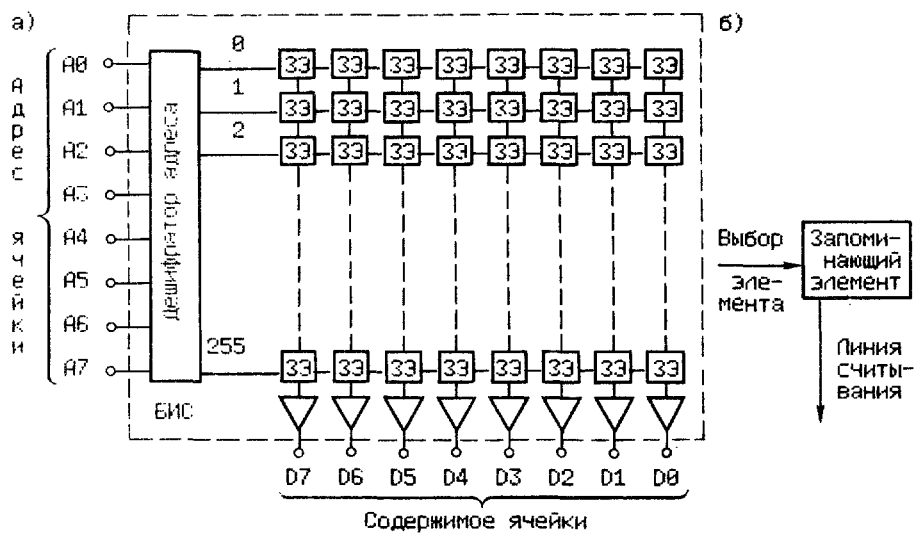


Рис. 6.2. ПЗУ 256$\times$8 с одномерной адресацией 8-битовых слов

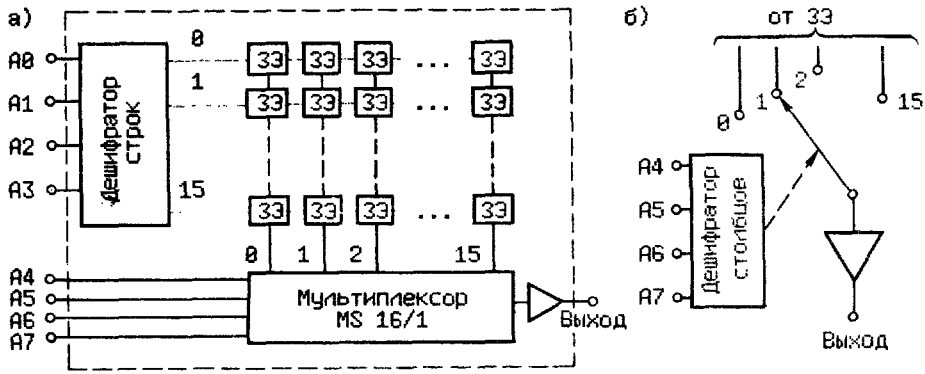


Рис. 6.3. ПЗУ 256×1 с двумерной адресацией 1-битовых слов

кальных линиях, соединяющих ЗЭ с усилителями считывания, появится код адресуемой ячейки, записанный в нее при изготовлении ПЗУ.

При реализации ЗУ с большим объемом памяти (большим числом адресов) резко возрастает количество выходов дешифратора адресов. Это усложняет изготовление БИС ЗУ. Для уменьшения числа выходов дешифратора используют двумерную (матричную) адресацию.

Дешифратор строк ПЗУ 256 × 1 (рис. 6.3) имеет всего 16 выходов, каждый из которых соединен с шестнадцатью ЗЭ, составляющими одну строку ПЗУ. Когда на одном из выходов дешифратора появляется единичный сигнал, на 16 вертикальных линий (столбцов) выводится содержимое ЗЭ, расположенных в соответствующей строке.

Электронный переключатель (мультиплексор) подключает к усилителю считывания один из шестнадцати ЗЭ. Выбор этого элемента осуществляется дешифратором столбцов, расположенным в мультиплексоре. Принцип работы мультиплексора иллюстрируется контактной моделью (рис. 6.3, *б*), в которой положение переключателя определяется выходным сигналом дешифратора столбцов. Для организации матричного ПЗУ 256 × 8 можно использовать восемь ПЗУ 256 × 1, расположенных в одной или восьми БИС. В одном из восьми ПЗУ 256 × 1 будут храниться биты с номером 0, в другом — с номером 1 и т. д.

Содержимое всех восьми выходов этих ПЗУ будет аналогично содержимому ПЗУ на рис. 6.2, *а*.

Следует отметить, что на практике при организации БИС ПЗУ для хранения многобитовых слов ЗЭ стараются размещать в матрицы с примерно одинаковым числом строк и столбцов. Поэтому ПЗУ 256 × 8 обычно имеет 32 строки (пятивходовой дешифратор строк) и 64 столбца. Первые восемь столбцов матрицы содержат старшие биты 8-разрядного слова, а последние восемь — младшие биты. Дешифраторы столбцов, расположенные в восьми мультиплексорах такого ПЗУ, имеют три входа, позволяющие адресовать один из восьми столбцов и выбрать содержимого одного из 32 ЗЭ, соединенных с этой линией.

При проектировании памяти микроЭВМ на БИС, как правило, не удается подобрать такую серийную БИС, которая содержала бы нужное число слов заданной разрядности: промышленность выпускает порядка десяти различных по емкости БИС ОЗУ с 1-битовыми и реже с 4-битовыми словами и несколько больше различных по емкости БИС ПЗУ (ППЗУ, СППЗУ) с 4-битовыми и 8-битовыми словами. Поэтому память обычно составляют из нескольких однотипных БИС.

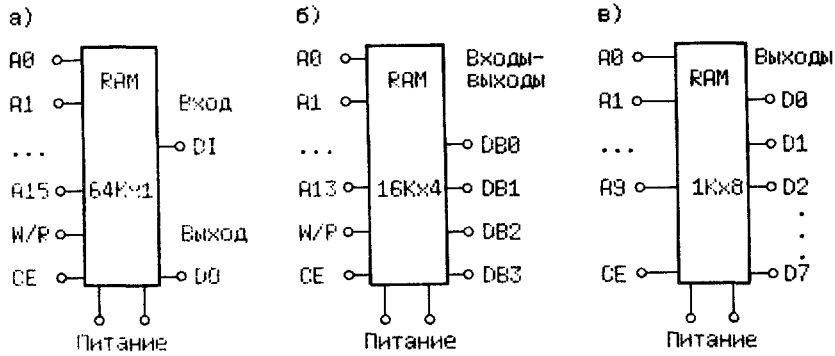


Рис. 6.4. Различные организации БИС памяти

Получение памяти со словами большей разрядности, чем разрядность слов БИС, не вызывает затруднений. Но как, например, при создании ПЗУ 512×8 из двух БИС ПЗУ 256×8 сообщить одной из них, что содержащееся на адресной шине число $(1\ 0000\ 0000)_2$ требует вывода 257-го, а не 1-го слова памяти, если на дешифраторы этих БИС подается одно и то же 8-разрядное число $(0000\ 0000)_2$?

Для этого БИС ЗУ снабжаются дополнительным входом «Выбор кристалла» (CE — **C**hip **E**nable). Добавив в блок ЗУ одновходовой дешифратор, соединив его вход со старшим разрядом адреса, а выходы — с входами CE БИС ПЗУ, мы разрешим выдачу данных с одной из БИС и перекроем выходы с другой БИС. При создании ПЗУ большего объема придется использовать дешифратор с бо́льшим числом входов, подключенных к соответствующим старшим разрядам адресной шины. Выходы этого дешифратора подключаются к БИС ПЗУ, обеспечивая выбор одной из них.

В БИС ОЗУ существует также вход для ее установки в режим записи или чтения данных (W/R — **W**rite/**R**ead) и могут быть отдельные (рис. 6.4, *а*) или совмещенные (рис. 6.4, *б*) выводы для ввода-вывода данных. Наибольшее распространение получили БИС ОЗУ с 1-битовыми словами. Таким БИС свойственны более простая внутренняя организация и меньшие габариты по сравнению с БИС той же информационной емкости, созданных для хранения многобитовых слов. Например, БИС ОЗУ 64К×1 должна иметь 16 адресных входов (2^{16}=64К), один вход и выход или один совмещенный вход-выход для передачи информации, т. е. 17—18 адресно-информационных выводов. Аналогичная по емкости БИС ОЗУ 8К×8 должна иметь 13 адресных входов (2^{13}=8К) и 16 раздельных или 8 совмещенных выводов для передачи информации, т. е. 21—29 адресно-информационных выводов.

БИС ПЗУ не содержат выводов для записи данных (эта запись выполняется при изготовлении БИС) и, следовательно, вывода W/R (рис. 6.4, *в*). БИС ППЗУ и СППЗУ программируются по линиям выход-вход при наличии сигнала записи и изображаются так же, как и БИС ОЗУ с многобитовыми словами (см. рис. 6.4, *б*).

Модули или банки памяти (отдельные крупные блоки большой памяти) составляют из набора однотипных БИС, связанных с системным интерфейсом ЭВМ через контроллер модуля памяти, который нередко называют интерфейсом модуля памяти. Рассмотрим один из возможных способов организации модуля ОЗУ емкостью 16К байт.

Массив БИС модуля памяти (рис. 6.5) состоит из 32 БИС 4К×1, которые организованы в матрицу из восьми строк и четырех столбцов. Адресные входы всех БИС объединены между собой и соединены с линиями *A0—A11*

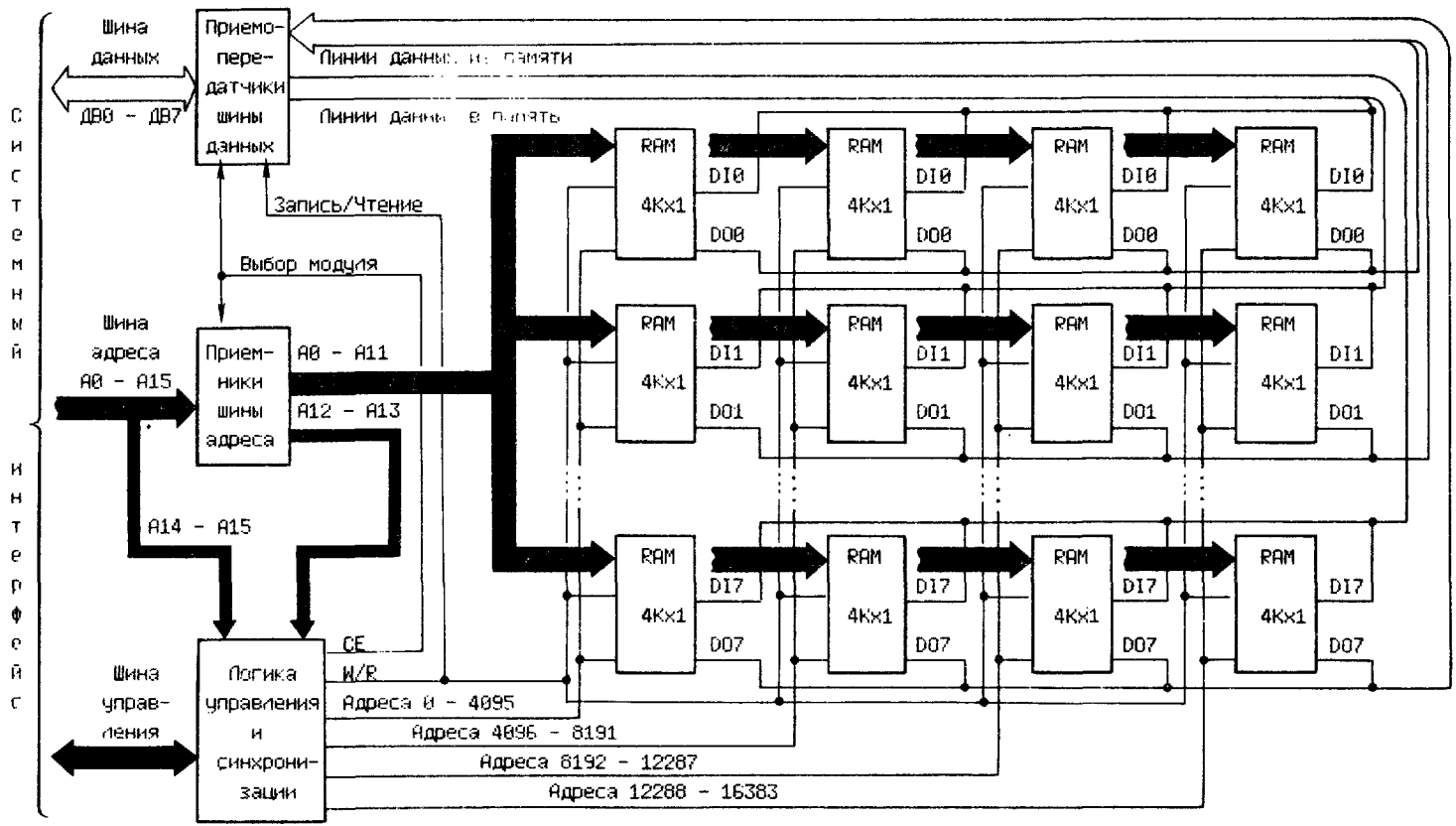


Рис. 6.5. Организация модуля ОЗУ 16 Кбайт из 32 БИС 4К×1

шины адреса. Объединены также входы W/R (Запись/Чтение). Однако по входам *CE* (Выбор кристалла) БИС разделены на четыре группы — четыре столбца. Выбор одного из столбцов осуществляется путем дешифрации кода адресных линий *A12* и *A13* в логике управления и синхронизации. Если модулю присвоены адреса от 0 до 16383 (подробнее ниже), то столбцы соответствуют ячейкам памяти 0—4095, 4096—8191, 8192—12287, 12288—16383. При поступлении в память, например, адреса $(0010\ 0000\ 0000\ 0000)_2$ и приказа «Чтение» логика управления вырабатывает сигналы R и CE2, разрешающие выдачу на приемопередатчики содержимого битов с адресом 0 из восьми БИС третьего столбца, т. е. 8-битового слова с адресом 8192.

А как реагирует модуль памяти на адреса, выходящие за пределы адресного пространства 0—16383, например: на адрес $(1010\ 0000\ 0000\ 0000)_2$, отличающийся от рассмотренного выше адреса лишь единицей в разряде *A15*?

Так как 16-разрядная шина позволяет адресовать 64К слов, то память ЭВМ может содержать еще три модуля, аналогичные рассматриваемому. Выбор одного из этих модулей определяется кодом, содержащимся в разрядах *A15* и *A14* адреса. Чтобы не делать еще одного общего контроллера памяти, в логику управления модуля памяти вводят компаратор (сравнивающее устройство).

Компаратор имеет две группы входов. Одну группу входов подключают к старшим разрядам адресной шины, а на другую задают (с помощью перемычек) такую комбинацию нулей и единиц, которая соответствует нужной границе начального адреса модуля. Если коды на этих входах совпадают, то логика управления и синхронизации вырабатывает сигнал *CE*, открывающий путь для прохождения адреса и данных, а также управляющего сигнала W/R и одного из сигналов выбора столбца CE0, CE1, CE2 или CE3. Для рассматриваемого модуля (см. рис. 6.5) компаратор настроен на комбинацию «00», т. е. на работу с адресами от 0 до 16383. При настройке компаратора на комбинацию «10» ячейки модуля памяти будут иметь адреса от 32768 до 49151.

Связь модуля с системным интерфейсом осуществляется через шинные (магистральные) приемопередатчики (формирователи). Это набор тристабильных вентилей, которые при наличии разрешающего управляющего сигнала создают на выходе высокий или низкий потенциал, соответствующий единичному или нулевому входному сигналу. При отсутствии разрешающего управляющего сигнала вентиль имеет очень высокое полное выходное сопротивление, т. е. фактически отключается от шины. Такие свойства позволяют производить обмен при минимальной электрической нагрузке на шину. Иногда эти элементы называют тристабильными усилителями мощности, буферными усилителями или драйверами. В приемопередатчиках буферы собраны парами для обеспечения ввода или вывода информации по сигналу, поступающему на вход W/R (см. рис. 6.5). Однако этот сигнал проходит лишь при разрешающем сигнале на входе CE.

Нетрудно заметить, почему память не может выдать содержимое адресуемой ячейки сразу же после пересылки в нее адреса и приказа на считывание информации, т. е. на что затрачивается время, названное в параграфе 6.1 временем доступа (см. рис. 6.1). Это время еще больше возрастает в модулях памяти микроЭВМ с мультиплексированной системной шиной адрес-данные (см. рис. 5.2, *в, д*). Такие модули сначала должны прочитать с шины и запомнить адрес требуемой ячейки (для этого в них устанавливается специальный буферный регистр адреса) и лишь после этого осуществить операции по выдаче или приему информации.

6.3. Оперативные и постоянные ЗУ с произвольным доступом

В настоящее время промышленность изготавливает два основных типа запоминающих элементов ОЗУ: статические и динамические запоминающие устройства.

Статический запоминающий элемент — это бистабильная схема (схема с двумя устойчивыми состояниями), обычно называемая триггером. В наиболее распространенных БИС ОЗУ, построенных на МОП (металл-окисел-полупроводник)-технологии, такие схемы выполнены на шести МОП-транзисторах (рис. 6.6, *а*) с обязательным использованием внешнего источника питания ($E_{пит}$). В них транзисторы *T3* и *T4* выполняют роль нагрузочных резисторов, а соединенные крест-накрест транзисторы *T1* и *T2* всегда (кроме момента переключения) находятся в противоположных состояниях. Когда один из транзисторов (например, *T2*) открыт (проводит ток), то с него на другой транзистор (*T1*) поступает запирающий потенциал. В свою очередь, с закрытого транзистора снимается потенциал, поддерживающий проводящее состояние открытого транзистора.

В таком устойчивом состоянии (которому при закрытом *T1* присвоено значение 0) схема находится до тех пор, пока с помощью отпирающего сигнала «Выбор элемента» не откроются транзисторы *T5*, *T6* и через них на *T1* и *T2* одновременно не поступят с линий W/R и $\overline{W/R}$ открывающий («1») и закрывающий («0») сигналы. После переключения схема опять будет находиться в устойчивом состоянии (которому присвоено значение 1) до нового разрешения (единичный сигнал «Выбор элемента») и записи нуля (0 на линии W/R и 1 на линии $\overline{W/R}$). Если же на элемент, сохраняющий 1, будет подан набор сигналов, соответствующий записи 1, то его состояние не изменится.

Динамический запоминающий элемент можно построить на трех МОП-транзисторах (рис. 6.6, *б*). В нем информация запоминается на паразитной емкости, всегда существующей между электродами транзистора (на схеме эта емкость показана в виде отдельного конденсатора $C_п$). Хранение данных в таком запоминающем элементе связано с состоянием проводимости *T2*, которое определяется зарядом конденсатора. Если заряд конденсатора обеспечивает достаточный открывающий потенциал, то *T2* открыт. Это состояние не является самоподдерживающимся, поскольку конденсатор постепенно саморазряжается. Если же заряд конденсатора мал или отсутствует, то *T2* не проводит и это состояние является самоподдерживающимся. Кроме *T2* в каждом запоминающем элементе присутствуют два транзистора для подключения элемента к линиям «Запись» и «Чтение».

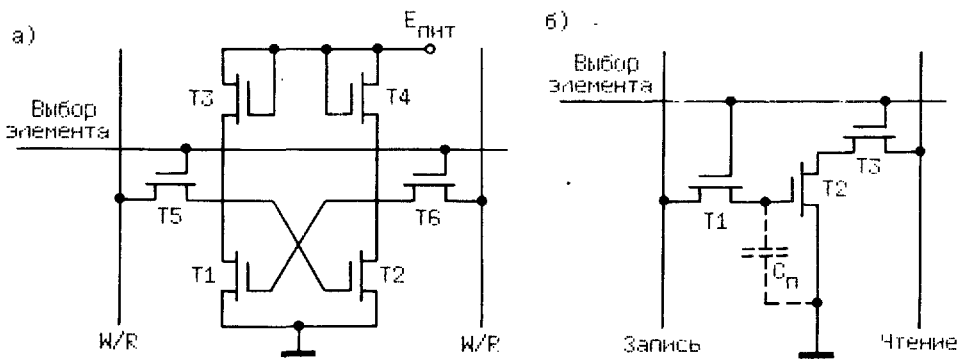


Рис. 6.6. Запоминающие элементы:
а — статического ОЗУ; *б* — динамического ОЗУ

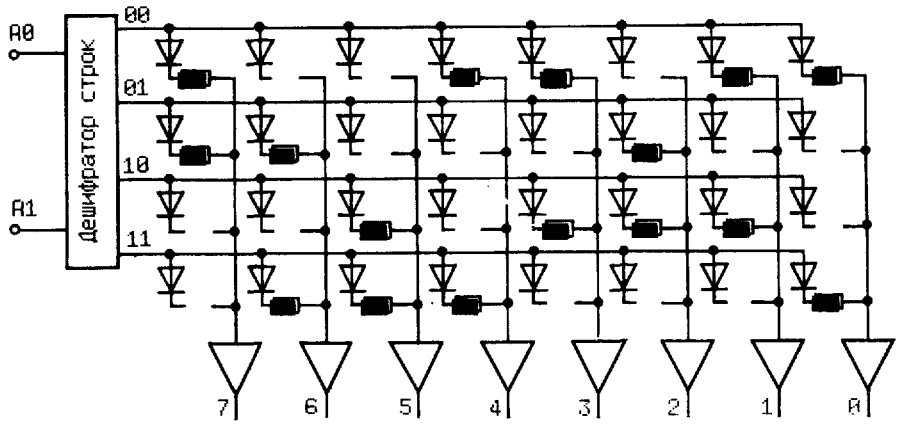
Из-за разряда запоминающих емкостей, вызванного током утечки, необходимо периодически (через несколько миллисекунд) подзаряжать емкости, хранящие заряд. Упрощения выполнения такой задачи можно добиться организацией перезаписи содержимого каждого элемента, не заботясь о том, была ли заряжена емкость этого элемента или нет. Тогда, связывая между собой линии «Чтение» и «Запись» каждого столбца и подавая периодически сигналы на линии выбора строк (опрашивая по линии «Выбор элемента» все элементы, расположенные в этой строке), обеспечивают перезапись каждого из элементов строки (содержимое, появившееся на линии «Чтение» какого-либо столбца, поступает через формирователь на линию «Запись» этого столбца и далее в тот же элемент, с которого считывалась информация). Такой процесс называется регенерацией.

В большинстве БИС динамических ОЗУ регенерация всех элементов памяти должна выполняться через 1—2 мс. Для этого модули динамической памяти снабжают логической схемой регенерации, которая реализует наиболее приемлемую стратегию такого процесса. В простейшем случае схема регенерации может выработать и подать на дешифраторы строк БИС адрес первой строки, а также сигналы «Выбор кристалла» всех БИС, т. е. одновременно регенерировать элементы, расположенные в первых строках всех БИС модуля, затем выработать аналогичные сигналы обращения ко второй строке всех БИС и т. д., пока не будут регенерированы все строки (все элементы памяти).

Так как количество строк БИС динамических ОЗУ и цикл памяти их элементов позволяют закончить такой процесс за несколько десятков микросекунд, то на регенерацию памяти затрачивается не более 1—2 % всего времени ее работы. Однако в процессе регенерации память недоступна для других устройств, и поэтому, например, исполнение запроса процессора на запись или чтение может задержаться на несколько десятков микросекунд. Если такой режим работы нежелателен, то логическая схема регенерации снабжается дополнительными блоками, позволяющими приостановить процесс регенерации на время выполнения запроса процессора или другого устройства, связанного с памятью.

Следовательно, модули, построенные на БИС динамических ОЗУ, имеют более сложные схемы управления, чем модули, построенные на БИС статических ОЗУ. Кроме того, они могут задерживать прием или выдачу данных на сравнительно большие промежутки времени. Чем же компенсируются эти недостатки? Главное достоинство БИС динамических ОЗУ — это более высокая плотность упаковки информации. Меньшее число МОП-транзисторов в запоминающем элементе (не более трех вместо шести) позволяет разместить на одном кристалле большее число запоминающих элементов и уменьшить число БИС в модуле памяти. Второе преимущество связано с тем, что динамический запоминающий элемент не потребляет тока, за исключением тех относительно коротких отрезков времени, когда к нему обращаются. Третье достоинство, являющееся следствием двух первых,— меньшая стоимость, меньшие габариты и бóльшая надежность динамических ОЗУ по сравнению со статическими.

В заключение отметим основной недостаток полупроводниковых ОЗУ (статических и динамических) — это энергозависимость. Хранимая информация теряется даже при кратковременных неисправностях питания. Применение резервных источников питания (например, аккумуляторов) при отключении основного блока питания микроЭВМ пока не может решить проблемы энергонезависимости. Поэтому во всех ЭВМ используются не только ОЗУ, но и постоянные запоминающие устройства, которые не теряют информации при выключении питания и могут ее выдавать через тысячи и



СОДЕРЖИМОЕ П З У

Адрес	Сохраняемое значение	
00	1001 1011	(9B)
01	1100 0100	(C4)
10	0010 1110	(2E)
11	0111 0001	(71)

Рис. 6.7. ПЗУ емкостью 4 байт

десятки тысяч часов после записи. Правда, запись информации в такие устройства выполняется, как правило, вне ЭВМ, а ЭВМ может лишь читать записанную информацию. Но так можно хранить и использовать микропрограммы устройства управления процессора и других устройств ЭВМ, программы функционирования управляющих ЭВМ, наиболее употребительные программы универсальных ЭВМ и т. п. Существуют четыре типа постоянных запоминающих устройств различного назначения.

Постоянные запоминающие устройства с масочным программированием (ПЗУ) – это устройства, в которые информация записана раз и навсегда в процессе изготовления полупроводниковых БИС. Запись производится путем металлизации промежутков, позволяющих соединить через диоды (МОП-транзисторы) соответствующие линии строк и столбцов (рис. 6.7). Это делается с помощью маскирующих фотошаблонов, задающих участки металлизации, требующиеся для кодирования той или иной информации. Этим способом изготавливают ПЗУ для преобразования двоичного кода в коды символов (русских и латинских букв, цифр и т. п.), ПЗУ микропрограмм, программ управляющих микроЭВМ и т. д.

Программируемые постоянные запоминающие устройства (ППЗУ) отличаются от ПЗУ тем, что при их изготовлении все диоды соединяются с соответствующими столбцами с помощью плавких перемычек, т. е. по всем адресам ППЗУ с байтовой организацией записываются при изготовлении коды $(1111\ 1111)_2 = (FF)_{16}$. Программирование такого ППЗУ заключается в том, что на него последовательно подаются адреса слов и импульсами тока разрушаются перемычки в тех местах, где они не нужны. Если при программировании была допущена ошибка, то она не может быть исправлена (БИС выбрасывается, и производится программирование новой БИС).

Стираемые программируемые (репрограммируемые), постоянные запоминающие устройства (СППЗУ) позволяют производить запись и стирание информации. Организация СППЗУ отличается от организации ППЗУ тем,

что между линиями строк и столбцов установлены не диоды с плавкими перемычками, а специальные МОП-транзисторы (с изолированным затвором). После изготовления все МОП-транзисторы обладают очень большим сопротивлением (т. е. закрыты). Подачей импульса большой амплитуды МОП-транзистор переводится в проводящее состояние, которое он может сохранять даже по истечении 10 лет. Для возвращения МОП-транзисторов в исходное (закрытое) состояние их надо подвергнуть длительному воздействию ультрафиолетовых лучей. Групповое облучение всех МОП-транзисторов производится на специальных установках в течение 10—30 мин через прозрачное (кварцевое) окно в корпусе БИС. После этого БИС СППЗУ оказывается в исходном состоянии и ее можно снова программировать.

Электрически изменяемые постоянные запоминающие устройства (ЭИППЗУ) отличаются от СППЗУ тем, что в них после программирования можно вернуть в исходное состояние (стереть) любой отдельно взятый МОП-транзистор. Поэтому ЭИППЗУ скорее похожи на ОЗУ, у которых цикл записи в несколько тысяч раз больше цикла чтения. Кроме того, ЭИПЗУ энергонезависимы, однако они не обеспечивают столь длительного хранения информации, как СППЗУ.

6.4. Надежность запоминающих устройств

Отдельные БИС памяти могут исправно работать в течение десятилетий. Однако когда в памяти объединяются сотни или даже тысячи таких БИС, среднее время между сбоями, возникающими то в одном, то в другом кристалле, может составлять уже всего несколько часов. Для микроЭВМ, управляющей каким-либо процессом, такая частота сбоев недопустима.

Как возникают ошибки? В наиболее распространенных устройствах полупроводниковой памяти 0 и 1 представляются наличием или отсутствием отрицательного электрического заряда в тех областях кремниевого кристалла, физические свойства которых превращают их в потенциальные ямы или электронные ловушки для отрицательных зарядов. Когда в запоминающем элементе должен храниться 0, соответствующая ему потенциальная яма заполняется электронами; когда же в него нужно записать 1, яма освобождается от электронов. Затем, при считывании, заряд запоминающего элемента измеряется. Если он превосходит определенное пороговое значение, хранящийся в запоминающем элементе бит информации интерпретируется как 0, в противном случае — как 1.

Когда потенциальная яма каким-то образом утратит свой заряд или незаряженный запоминающий элемент случайно приобретет достаточно большой заряд, то значение хранящегося в нем бита станет ошибочным. Такие ситуации действительно возникают время от времени.

В случае повреждения микросхемы ошибка становится неустранимой. При этом тот или иной запоминающий элемент оказывается ненадежным — он либо застревает на каком-то одном значении, либо беспорядочно колеблется от одного значения к другому. Если повреждены адресные линии, то ненадежными может стать целая строка или столбец данных. Иногда микросхемы выходят из строя частично или полностью и их надо заменять.

Однако ошибки возникают и в неповрежденных микросхемах. Обычно виновником таких ошибок является альфа-частица — ядро атома гелия (состоящего из двух протонов и двух нейтронов), испущенное ядром тяжелого атома в результате радиоактивного распада. Альфа-частицы — отрицательный фактор, как правило, присущий устройствам памяти современных ЭВМ, поскольку радиоактивные ядра в небольших количествах присутствуют почти во всех материалах. В частности, атомы урана и тория содержатся

в пластмассовых корпусах обычных микросхем памяти. Таким образом, кристалл подвергается постоянной бомбардировке.

Когда альфа-частица, обладающая достаточно высокой энергией, проникает через потенциальную яму, электроны покидают те места, которые они прежде занимали в кристаллической решетке кремния, и могут попасть в близлежащие потенциальные ямы. Если яма была заполнена электронами (т. е. если в запоминающем элементе хранился 0), то это не приводит ни к каким последствиям. Если же потенциальная яма первоначально была «пустой» (т. е. в запоминающем элементе хранилась 1), то может возникнуть ошибка: 1 может превратиться в 0. Это так называемый сбой (случайная ошибка, появление которой можно предвидеть, но нельзя предотвратить).

Часто ли возникают такие ошибки? Чтобы получить ответ на этот вопрос, рассмотрим некое гипотетическое устройство памяти емкостью 1 Мбайт, состоящее из 128 БИС по 64 Кбит. Поскольку в каждой БИС содержатся 65 536 запоминающих элементов, память емкостью в 1 Мбайт имеет 8 388 608 запоминающих элементов. (По нынешним временам это совсем небольшая память.) Данные хранятся в виде слов по 16 бит в каждом.

Как долго может функционировать эта память без сбоев? Еще в эпоху, предшествующую появлению ЭВМ, статистики ввели понятие среднего времени до появления сбоя. Это промежуток времени, в течение которого в среднем устройство работает безошибочно. Среднее время до возникновения ошибки для одного запоминающего элемента в БИС, к счастью, очень велико. Лабораторные эксперименты (в которых во внимание принимались только ошибки устранимого типа) показали, что обычно значение этого показателя превосходит миллион лет.

Однако, как уже отмечалось, память ЭВМ может состоять из нескольких миллионов запоминающих элементов, а это означает, что в целом для запоминающего устройства среднее время до возникновения ошибки не так уж и велико. Предположим, что среднее время до появления ошибки для одного запоминающего элемента составляет ровно миллион лет. Тогда среднее время до появления ошибки в устройстве памяти емкостью в 1 Мбайт будет равно миллиону лет, поделенному на 8 388 608, и составит приблизительно 43 дня. Не так уж много! А достаточно экономичного способа защиты памяти от альфа-частиц просто нет.

Решить проблему помогли математики, которые, разрабатывая методы передачи данных в сложных телекоммутационных системах, еще 40 лет назад пришли к выводу, что в некоторых ситуациях лучше исправлять уже возникшие ошибки, чем пытаться их предотвратить.

Сначала они предложили добавлять к битам передаваемого или хранимого N-разрядного слова еще один дополнительный бит — бит четности (см. параграф 7.1). Содержимое этого бита подбирается таким образом, чтобы среди получившихся $N+1$ разрядов обязательно было четное число единиц. Однако такой избыточный код позволяет обнаружить лишь одиночную ошибку (если исчезают две единицы, то четность не нарушается и ошибка не обнаруживается). Добавлением K битов четности и соответствующим расположением их среди битов N-разрядного слова можно построить $N+K$-разрядный код, позволяющий обнаруживать место, в котором произошла ошибка и, следовательно, исправлять ее.

Подобные коды были впервые рассмотрены в 1948 г. Р. Хеммингом и с тех пор обычно называются кодами Хемминга. В них биты $N+K$-разрядного слова нумеруются слева направо начиная с 1, а не с 0. Все биты, номера которых равны степени числа 2, являются битами четности, а все оставшиеся биты используются как информационные. Так, к 16-разрядному информационному слову добавляют еще пять битов четности, располагая их в 1, 2, 4, 8

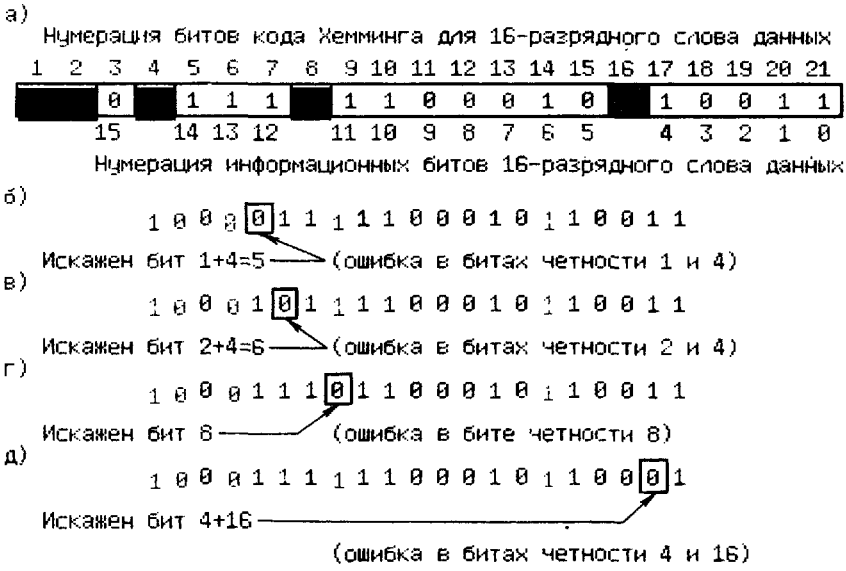


Рис. 6.8. Исправляющий код для 16-битового слова данных

Цветом — контрольные биты, а в черных рамках — искаженные биты

и 16 разрядах слова, а в разрядах 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20 и 21 размещают биты данных (рис. 6.8, *а*).

Каждый бит четности используется для контроля лишь определенных разрядов расширенного слова. Номера контролируемых разрядов для каждого бита четности приведены в табл. 6.1. Подчеркнем, что в число контролируемых разрядов включается и тот разряд, где расположен сам бит четности. При этом содержимое бита четности устанавливается так, чтобы суммарное число единиц в контролируемых им разрядах было четным.

Отметим также, что с помощью пяти битов четности можно контролировать 31 разряд, из которых 26 будут информационными разрядами. Поэтому такой код Хемминга обозначается как код (31, 26). Но так как в архитектуре современных ЭВМ приняты слова длиной 8, 16, 32 бит, то пять битов четности защищают меньшее количество данных, чем могли бы, т. е. всего 16 бит.

Рассмотрим теперь несколько примеров такого способа защиты сохраняемых данных.

Сначала проверим код числа, приведенного на рис. 6.8, *а*. В нем биты 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 и 21 содержат шесть единиц, т. е. бит четности 1 — правилен; биты 2, 3, 6, 7, 10, 11, 14, 15, 18 и 19 содержат четыре единицы, т. е.

Таблица 6.1

Номера битов четности и контролируемых ими разрядов слов в коде Хеммиига

Контрольные разряды	Контролируемые разряды
1	1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33...
2	2 3 6 7 10 11 14 15 18 19 22 23 26 27 30 31 34...
4	4 5 6 7 12 13 14 15 20 21 22 23 28 29 30 31 36...
8	8 9 10 11 12 13 14 15 24 25 26 27 28 29 30 31 40...
16	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 48...
32	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48...
...	...

бит четности 2 — правилен; биты 4, 5, 6, 7, 12, 13, 14, 15, 20 и 21 содержат шесть единиц, т. е. бит четности 4 — правилен; биты 8, 9, 10, 11, 12, 13, 14 и 15 содержат четыре единицы, т. е. бит четности 8 — правилен; биты 16, 17, 18, 19, 20 и 21 содержат четыре единицы, т. е. бит четности 16 тоже правилен.

Что же произойдет, если из-за проникновения альфа-частицы исчезнет единица в пятом разряде этого числа (рис. 6.8, *б*). Проверка образовавшегося кода дает следующий результат:

бит четности 1 неправилен (биты 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 и 21 содержат пять единиц);

бит четности 2 правилен (биты 2, 3, 6, 7, 10, 11, 14, 15, 18 и 19 содержат четыре единицы);

бит четности 4 неправилен (биты 4, 5, 6, 7, 12, 13, 14, 15, 20 и 21 содержат пять единиц);

бит четности 8 правилен (биты 8, 9, 10, 11, 12, 13, 14 и 15 содержат четыре единицы);

бит четности 16 правилен (биты 16, 17, 18, 19, 20 и 21 содержат четыре единицы).

Получение неправильного значения бита четности 1 указывает на то, что ошибка должна быть в одном из контролируемых им битов: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 или 21. Поскольку бит четности 2 правилен, то правильны и контролируемые им нечетные биты 3, 7, 11, 15 и 19, так что ошибка произошла не в них. Правильность контрольного бита 8 исключает возникновение ошибки в битах 9 и 13, а правильность контрольного бита 16 — в битах 17 и 21. Следовательно, под подозрением остаются биты 1 и 5. Так как неправилен и бит четности 4, который не контролирует бита с номером 1, но контролирует бит с номером 5, то можно сделать вывод об ошибочности бита 5. Инвертирование этого бита (изменение его значения с 0 на 1) исправляет положение — все биты четности становятся правильными.

Если проанализировать аналогичным образом коды на рис. 6.8, *в, г, д,* являющиеся искаженными значениями кода на рис. 6.8, *а*, то можно убедиться в правильности простого способа поиска и исправления испорченного бита:

1) произвести проверку всех битов четности;

2) если биты четности правильны, то перейти к п. 5;

3) вычислить сумму номеров всех неправильных битов четности;

4) инвертировать содержимое бита, номер которого равен сумме, вычисленной в п. 3;

5) передать правильный код числа, исключив из него все биты четности.

Теперь можно вернуться к вопросу о надежности устройства памяти емкостью 1 Мбайт. Применение корректирующего кода Хемминга означает, что на самом деле память объемом 1 Мбайт не может состоять лишь из 128 БИС по 64 Кбит. Она должна состоять из 168 таких БИС, так как каждое слово памяти содержит уже 21 бит, а не 16. Пять лишних битов в каждом слове — это биты четности. Они недоступны пользователю, так как зарезервированы для образования корректирующего кода.

Насколько большей надежностью отличается память с корректирующим кодом по сравнению с памятью, в которой он не применяется? С одной стороны, увеличение количества запоминающих элементов, связанное с введением битов четности, означает некоторое увеличение частоты возникновения ошибок. Среднее время до возникновения ошибки составит 16/21 от прежнего значения или около 33 дней. С другой стороны, теперь каждое кодовое слово длиной 21 бит защищено от одиночной ошибки в любом из своих разрядов. Сбой происходит только тогда, когда в кодовом слове возникают две ошибки.

В самом деле, первая ошибка не вызывает никаких проблем. Если она возникает, ее можно сразу исправить. Вторая по счету ошибка также не вызывает затруднений, если только она не возникла в том же кодовом слове, что и первая. Третья ошибка приводит к сбою только в том случае, если она возникает или в первом, или во втором кодовом слове, которые уже содержат по одной ошибке, и т. д. Поскольку в нашем случае содержимое памяти включает 524 288 кодовых слов, пройдет очень много времени, прежде чем в одном и том же кодовом слове возникнут две ошибки: среднее время до возникновения сбоя составит

$$33\sqrt{(\pi/2)\ 524288/356} = 82 \text{ года.}$$

Короче говоря, если не пользоваться корректирующим кодом, среднее время до возникновения сбоя составляет несколько десятков дней, если же им пользоваться, то оно увеличивается до нескольких десятков лет (почти в 1000 раз). Это соотношение служит прекрасной иллюстрацией эффективности метода коррекции ошибок.

Глава 7

ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА МИКРОЭВМ

7.1. Общие сведения о периферийных устройствах и их связи с микроЭВМ

Классификация периферийных устройств. Появление сравнительно недорогих микропроцессоров и микроЭВМ, построенных на их основе, дало возможность: управлять экспериментом и технологическим процессом; производить сбор данных и выполнять расчеты непосредственно под управлением вычислительной машины; управлять исключающим занос торможением автомобиля, бытовыми приборами и машинами; иметь дома персональную ЭВМ и использовать ее для хранения кулинарных рецептов, как противника в играх, в качестве обучающей системы и секретаря. Для этого микроЭВМ должна быть связана с окружающей средой через периферийные (внешние) устройства. Такие устройства можно подразделить на три группы:

устройства для связи человек — микроЭВМ (различные клавиатуры, индикаторы, устройства для работы с перфолентами, электрические пишущие машинки, символьные и графические дисплеи, графопостроители, читающие автоматы, устройства речевого обмена информацией и т. п.);

устройства для связи с объектами управления (различные датчики и исполнительные органы, а также устройства для преобразования непрерывных сигналов с датчиков в цифровые сигналы и обратного преобразования при выдаче информации на исполнительные органы);

внешние запоминающие устройства большой емкости (ЗУ на магнитных лентах и дисках, память на цилиндрических магнитных доменах и приборах с зарядовой связью).

Кодирование информации. Так как большинство внешних устройств (ВУ) позволяет человеку общаться с ЭВМ на привычном ему языке слов и десятичных чисел, а ЭВМ понимает лишь язык физических состояний, кодируемых двоичными числами, то в ВУ обычно производится кодирование

(декодирование) информации, пересылаемой в (из) микроЭВМ. Для кодирования русских (33 буквы) и латинских (26 букв) прописных и строчных букв, а также различных графических (например, символа %) и служебных (например, код для перевода строки в пишущей машинке) символов необходимо иметь более чем $2^7 = 128$ различных кодов. Поэтому кодирование обычно производится в 8-битовом (1-байтовом) коде обмена информацией КОИ-8 (см. табл. Б.1 и Б.2 приложения Б).

В соответствии с ГОСТ 13052–74 существуют также 7-разрядные коды: КОИ-7H_0, КОИ-7H_1 (по восемь колонок) и КОИ-7C_1 (две колонки). Таблицы этих кодов составлены из колонок таблицы КОИ-8: КОИ-7H_1 из восьми первых колонок табл. Б.1, КОИ-7H_0 из четырех первых и четырех последних колонок табл. Б.1 и КОИ-7C_1 из колонок с номерами 8 и 9 табл. Б.1.

Таким образом, с помощью КОИ-7H_1 кодируются символы и латинские буквы, с помощью КОИ-7H_0 — символы и русские буквы, а с помощью КОИ-7C_1 — дополнительные управляющие символы. Для перехода с КОИ-7H_0 на КОИ-7H_1 надо послать код символа ВЫХ, а для обратного перехода — код символа ВХ. Каждый символ КОИ-7C_1 должен предваряться кодом символа АР2.

Пример 7.1. Закодировать слова ЧИЖ и CROW (ворона).

При использовании кода КОИ-8 получим:

$$(\text{FE E9 F6})_{16} = (11111110\ 11101001\ 11110110)_2$$

и

$$(\text{43 52 4F 57})_{16} = (01000011\ 01010010\ 01001111\ 01010111)_2.$$

При использовании кода КОИ-7H_0 для слова ЧИЖ и кода КОИ-7H_1 для слова CROW получим:

$$(\text{7E 69 76})_{16} = (1111110\ 1101001\ 1110110)_2$$

и

$$(\text{43 52 4F 57})_{16} = (1000011\ 1010010\ 1001111\ 1010111)_2.$$

Так как ВУ имеют достаточно много механических элементов, то они менее надежны, чем электронные схемы процессора и памяти. Причиной неправильного восприятия кода считываемого символа могут быть пыль на считывающих головках накопителей на магнитных лентах или дисках, царапины на самих лентах и дисках, масляные пятна на старых перфолентах и т. п. Искажения из-за электрических помех возникают и при передаче данных между процессором (памятью) и ВУ (контроллером ВУ). Короче говоря, если надо ввести, вывести или передать символы, то могут появиться ошибки. Для обнаружения таких ошибок (а иногда и для их исправления) коды символов преобразуют в какую-либо стандартную форму (например, добавляют еще один бит, чтобы в новом коде было четное число единиц). При передаче такого стандартного кода производится определенного вида контроль (например, контроль четности), и по его результатам принимается решение об использовании полученного символа.

Пример 7.2. Закодировать слово ЧИЖ, используя 8-разрядный четный код, полученный из КОИ-7H_0.

Кодируемый символ	Бит четности	КОИ-7H_0 2^6 2^5 2^4 2^3 2^2 2^1 2^0	Количество единиц в коде
Ч	0	1 1 1 1 1 1 0	6
И	0	1 1 0 1 0 0 1	4
Ж	1	1 1 1 0 1 1 0	6

Если во время передачи ошибочно изменится значение одного бита, то число единиц в полученном символе станет нечетным и получатель узнает, что произошла ошибка.

Способы обмена данными между ВУ и микроЭВМ. Теперь рассмотрим некоторые общие вопросы, связанные с обменом данными между ВУ и микроЭВМ. Существуют два способа передачи слов информации по линиям данных:

параллельный, когда одновременно пересылаются все биты слова;

последовательный, когда биты слова пересылаются поочередно, начиная, например, с его младшего разряда.

Так как между отдельными проводниками шины для параллельной передачи данных существует электрическая емкость, то при изменении сигнала, передаваемого по одному из проводников, возникает помеха (короткий выброс напряжения) на других проводниках. С увеличением длины шины (увеличением емкости проводников) помехи возрастают и могут восприниматься приемником как сигналы. Поэтому рабочее расстояние для шины параллельной передачи данных ограничивается длиной 1—2 м, и только за счет существенного удорожания шины или снижения скорости передачи длину шины можно увеличить до 10—20 м.

Указанное обстоятельство и желание использовать для дистанционной передачи информации телеграфные и телефонные линии обусловили широкое распространение способа последовательного обмена данными между ВУ и микроЭВМ и между несколькими микроЭВМ.

Возможны два режима последовательной передачи данных: синхронный и асинхронный.

При синхронной последовательной передаче каждый передаваемый бит данных сопровождается импульсом синхронизации, информирующим приемник о наличии на линии информационного бита. Следовательно, между передатчиком и приемником должны быть протянуты минимум три провода: два для передачи импульсов синхронизации и битов данных, а также общий заземленный проводник. Если же передатчик (например, микроЭВМ) и приемник (например, дисплей) разнесены на несколько метров, то каждый из сигналов (информационный и синхронизирующий) придется посылать либо по экранированному (телевизионному) кабелю, либо с помощью витой пары проводов, один из которых заземлен или передает сигнал, инверсный основному.

Синхронная последовательная передача начинается с пересылки в приемник одного или двух символов синхронизации (не путать с импульсами синхронизации). Получив такой символ (символы), приемник начинает прием данных и их преобразование в параллельный формат. Естественно, что при такой организации синхронной последовательной передачи она целесообразна лишь для пересылки массивов слов, а не отдельных символов. Это обстоятельство, а также необходимость использования для обмена сравнительно дорогих (четырехпроводных или кабельных) линий связи помешало широкому распространению синхронной последовательности передачи данных.

Асинхронная последовательная передача данных означает, что у передатчика и приемника нет общего генератора синхроимпульсов и что синхронизирующий сигнал не посылается вместе с данными. Как же в таком случае приемник будет узнавать о моментах начала и завершения передачи битов данных? Опишем простую процедуру, которую можно использовать, если передатчик и приемник асинхронной последовательной передачи данных согласованы по формату и скорости передачи.

Стандартный формат асинхронной последовательной передачи данных, используемый в ЭВМ и ВУ, содержит n пересылаемых битов информации (при пересылке символов n равно 7 или 8 битам) и 3—4 дополнительных бита: стартовый бит, бит контроля четности (или нечетности) и 1 или 2 сто-

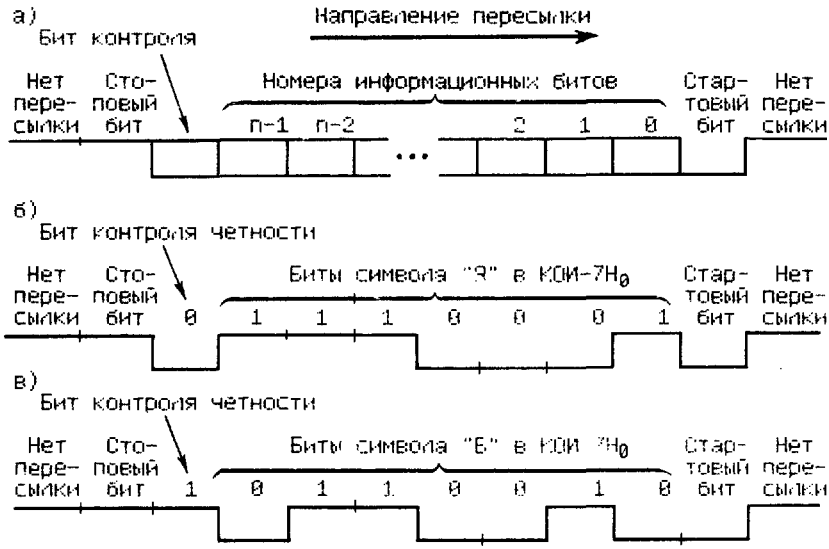


Рис. 7.1. Формат асинхронной последовательной передачи данных

повых бита (рис. 7.1, *а*). Когда передатчик бездействует (данные не посылаются в линию), на линии сохраняется уровень сигнала, соответствующий логической 1.

Передатчик может начать пересылку символа в любой момент времени посредством генерирования стартового бита, т. е. перевода линии в состояние логического 0 на время, точно равное времени передачи бита. Затем происходит передача битов символа начиная с младшего значащего бита, за которым следует дополнительный бит контроля по четности или нечетности. Далее с помощью стопового бита линия переводится в состояние логической 1 (рис. 7.1, *б*). При единичном бите контроля стоповый бит не изменяет состояния сигнала на линии (рис. 7.1, *в*). Состояние логической 1 должно поддерживаться в течение промежутка времени, равного 1 или 2 временам передачи бита.

Промежуток времени от начала стартового бита до конца стопового бита (стоповых битов) называется кадром. Сразу после стоповых битов передатчик может посылать новый стартовый бит, если имеется другой символ для передачи; в противном случае уровень логической 1 может сохраняться на протяжении всего времени, пока бездействует передатчик. Новый стартовый бит может быть послан в любой момент времени после окончания стопового бита, например через промежуток времени, равный 0,43 или 1,5 времени передачи бита.

В линиях последовательной передачи данных передатчик и приемник должны быть согласованы по всем параметрам формата, изображенного на рис. 7.1, включая номинальное время передачи бита. Для этого в приемнике устанавливается генератор синхроимпульсов, частота которого должна совпадать с частотой аналогичного генератора передатчика. Кроме того, для обеспечения оптимальной защищенности сигнала от искажения, шумов и разброса частоты синхроимпульсов приемник должен считывать принимаемый бит в середине его длительности.

Рассмотрим работу приемника с того момента, когда он окончил прием символа данных и перешел в режим обнаружения стартового бита следующего слова.

Если линия перешла в состояние логического нуля и находится в этом состоянии в течение времени, не меньшего половины временнóго интервала передачи бита, то приемник переводится в режим считывания битов информации. В противном случае приемник остается в режиме обнаружения, так как вероятнее всего это был не стартовый бит, а шумовая помеха. В новом режиме приемник вырабатывает сигналы считывания через интервалы, равные времени передачи бита, т. е. выполняет считывание и сохранение принимаемых битов примерно на середине их передачи. Аналогичным образом будут считаны бит контроля по четности и сигнал логической единицы (стоповый бит). Если оказалось, что на месте стопового бита обнаружен сигнал логического нуля, то произошла «Ошибка кадра» и символ принят неправильно. Иначе проверяется, четно ли общее число единиц в информационных битах и бите контроля, и если оно четно, производится запись принятого символа в буфер приемника.

Передний фронт стартового бита сигнализирует о начале поступления передаваемой информации, а момент его появления служит точкой отсчета времени для считывания битов данных. Стоповый бит предоставляет время для записи принятого символа в буфер приемника и обеспечивает возможность выявления ошибки кадра. Наиболее часто ошибки кадра появляются тогда, когда приемник ошибочно синхронизирован с битом 0, который в действительности не является стартовым битом. Если передатчик бездействует (посылает сигнал логической единицы) в течение одного кадра или более, то всегда можно восстановить правильную синхронизацию. Хуже обстоит дело при рассинхронизации генераторов передатчика и приемника, когда временнóй интервал между сигналами считывания принимаемых битов будет меньше или больше времени передачи бита.

Например, если при считывании битов посылки, показанной на рис. 7.1, *б*, временнóй интервал между сигналами считывания станет на 6 % меньше, чем время передачи бита, то восьмой и девятый сигналы считывания будут выработаны тогда, когда на линии находится бит контроля четности (рис. 7.2). Следовательно, не будет обнаружен стоповый бит и будет зафиксирована ошибка кадра, несмотря на правильность принятой информации. Однако при 18 %-ной рассинхронизации генераторов, когда вместо кода $(01110001)_2$ приемник зафиксирует код $(11100001)_2$, никаких ошибок не будет обнаружено четность соблюдена и стоповый (девятый по порядку) бит равен 1 (см. рис. 7.2).

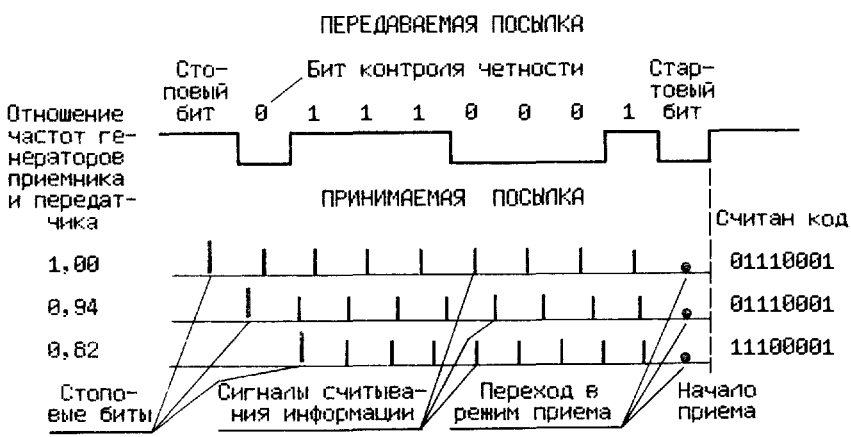


Рис. 7.2. Ошибки из-за рассинхронизации генераторов передатчика и приемника

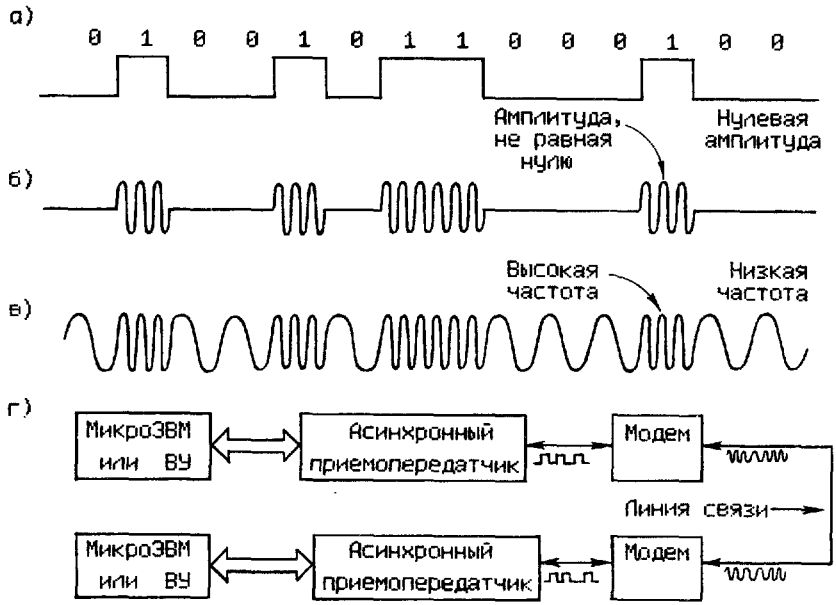


Рис. 7.3. Побитовая передача данных по линии связи

Дистанционная связь. Когда микроЭВМ должна обмениваться данными с удаленными ВУ или другими микроЭВМ (находящимися в другой комнате, другом конце здания или другом доме), то возникает вопрос: как и с помощью чего связать эти объекты. Относительно большое расстояние между объектами предопределяет последовательный (побитный) обмен данными. Это же расстояние обусловливает экономическую нецелесообразность использования для связи специальных кабельных линий и наводит на мысль об обмене через телефонные линии или радиоканал.

Телефонные линии предназначены для передачи человеческого голоса, а не цифровых данных. Двухуровневые сигналы (рис. 7.3, *а*) претерпевают значительные искажения во время их передачи по телефонным линиям, что приводит к изменению передаваемых данных. В то же время синусоидальный сигнал с частотой от 1000 до 2500 Гц, называемой несущей, может быть передан с относительно малыми искажениями. Поскольку значения простой синусоидальной волны полностью предсказуемы, то она сама по себе не передает никакой информации. Однако путем изменения ее амплитуды, частоты или фазы можно передавать последовательность нулей и единиц, как это показано на рис. 7.3. Такой процесс называют модуляцией.

При амплитудной модуляции (рис. 7.3, *б*) для представления значений 0 и 1 используются соответственно два различных уровня напряжения. Если послушать такую передачу цифровых данных с очень низкой скоростью передачи, то можно услышать громкий фон при 1 и не услышать никакого фона при 0. При частотной модуляции (рис. 7.3, *в*) уровень напряжения сохраняется постоянным, однако несущая частота принимает различные значения для 1 и 0. Если послушать передачу частотно-модулированных цифровых данных, то можно услышать два фона, соответствующих 0 и 1.

Таким образом, когда микроЭВМ должна быть связана с ВУ или другой ЭВМ по телефонному каналу (любой двухпроводной линии связи или радиоканалу), между ними следует установить устройства для преобразования данных из параллельного кода в последовательный код (двухуровневый

сигнал напряжения) и для обратного преобразования, а также для преобразования двухуровневого сигнала напряжения в амплитудно-, частотно- или фазово-модулированный сигнал и для обратного преобразования (рис. 7.3, *г*). Последнее устройство получило название модем (**мо**дулятор-**дем**одулятор).

Для надежной расшифровки модулированной информации передаваемые биты должны быть преобразованы в достаточно представительные части синусоидальной волны, частота которой ограничивается полосой пропускания линии связи (для телефонного канала полоса пропускания 3000 Гц). Это накладывает ограничения на скорость передачи цифровой информации по каналу связи. При частотной модуляции, когда для представления 0 и 1 используются токовые сигналы с частотами 1070 и 1270 Гц, скорость передачи информации не должна превышать 1200 бит/с. Для увеличения этой скорости применяют более сложные методы модуляции, при которых каждый передаваемый сигнальный элемент кодирует не один, а несколько битов. Это позволяет поднять скорость передачи по телефонной линии до 9600 бит/с, а по специальной линии — до 19,2К бит/с.

При передаче данных по линиям связи используются три режима: симплексный (от лат. simplex — простой), полудуплексный и дуплексный (от лат. duplex — двойной). Симплексная линия обеспечивает передачу данных только в одном направлении. Причиной этого являются не какие-то особенности проводов, а просто тот факт, что на одном конце линии имеется только передающее устройство, а на другом конце — только приемное. Полудуплексная связь обеспечивает передачу и получение информации в обоих направлениях, но неодновременно. Во время любой передачи один модем посылает, а другой получает. Дуплексная связь обеспечивает передачу и получение данных в обоих направлениях одновременно. Большинство модемов может обеспечить дуплексную связь с помощью одной телефонной линии. Это достигается путем использования другой пары частот для связи в обратном направлении, например: 2025 Гц — для представления 0 и 2225 Гц — для представления 1.

7.2. Устройства связи человек — микроЭВМ

Кнопки, переключатели и клавиатура — это простейшее устройство для ввода информации в микроЭВМ. С их помощью могут вводиться те или иные данные (программы), устанавливаться режимы работы микроЭВМ и управляемых ею устройств, передаваться команды и т. д.

Общая проблема, возникающая при использовании таких устройств ввода, состоит в том, что электромеханические переключатели не могут обеспечить четкого фронта сигнала замыкания (размыкания). В момент нажатия кнопки (переключения тумблера, клавиши и т. п.), стоящей в электрической цепи, происходит нерегулярное замыкание этой цепи до тех пор, пока не установится надежный контакт. Длительность этого периода зависит от конструкции кнопки (переключателя), но обычно не превышает 2 мс. Нестабильность сигнала («дребезг контакта») устраняют аппаратным (схемным) или программным способом. В последнем случае, когда микроЭВМ обнаруживает замыкание (размыкание) цепи, она начинает выполнять программный цикл задержки в течение времени, достаточного для прекращения скачков напряжения. Программа может быть составлена так, чтобы в это время могла выполняться другая работа.

При использовании многопозиционных переключателей (клавиатур) микроЭВМ или контроллер (см. параграфы 3.2 и 8.1) должны:

1) определить, изменилось ли положение переключателя (была ли нажата хотя бы одна клавиша);

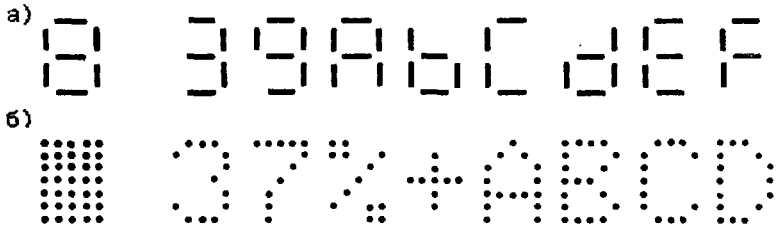


Рис. 7.4. Многоэлементные индикаторы:
а — 7-сегментный; *б* — точечная матрица 5×7

2) если положение переключателя изменилось (нажата одна из клавиш), определить, в каком положении он находится (какая клавиша нажата);

3) выбрать действие, соответствующее новому положению переключателя (нажатой клавише).

Для реализации этих операций часто используют специальные микросхемы шифраторов клавиатур, кодеров на ПЗУ и др. Существуют даже БИС контроллеров клавиатур.

Индикатор — отдельный светоизлучающий элемент (сигнальная лампочка, светодиод и т. п.) — является простейшим устройством вывода. С его помощью можно информировать о состоянии какого-либо устройства, подключенного к микроЭВМ, например, путем подсветки той или иной надписи на лицевой панели устройства или высвечиванием двоичного кода содержимого регистров устройства и т. п.

Гораздо удобнее использовать многоэлементные индикаторы, позволяющие выводить данные в привычной буквенно-цифровой форме. В таких индикаторах буквы, цифры и другие символы формируются на сегментном поле, состоящем из 7, 9 или 14 сегментов, или на точечной матрице 5×7, 7×9 или 10×14 (рис. 7.4). Последние позволяют формировать большее число символов.

В настоящее время промышленность стала выпускать большое число газоразрядных матричных панелей. Они представляют собой матрицу 32×32, 64×64, 128×128, 256×256 или 512×512 элементов (газоразрядных ячеек), каждый из которых может светиться достаточно ярким зеленым или красным цветом. С помощью таких панелей и наборов из таких панелей можно выводить достаточно большой объем информации, сформированной из отдельных светящихся точек.

Перфоленточные устройства ввода-вывода — одни из самых первых внешних устройств ЭВМ. Они достаточно широко применяются при подготовке данных к ряду устройств гибких автоматизированных производств (ГАП), использующих микроЭВМ.

Перфолента — длинная и узкая бумажная или пластмассовая лента, на которой информация кодируется пробивками круглых отверстий (рис. 7.5).

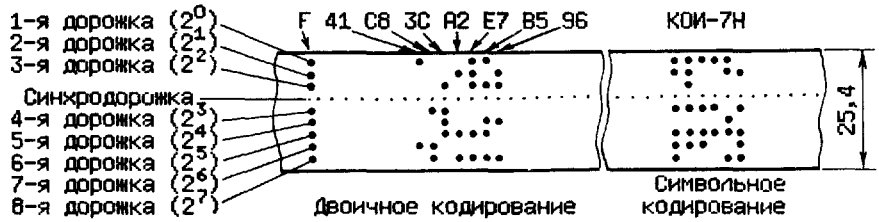


Рис. 7.5. Восьмидорожечная перфолента

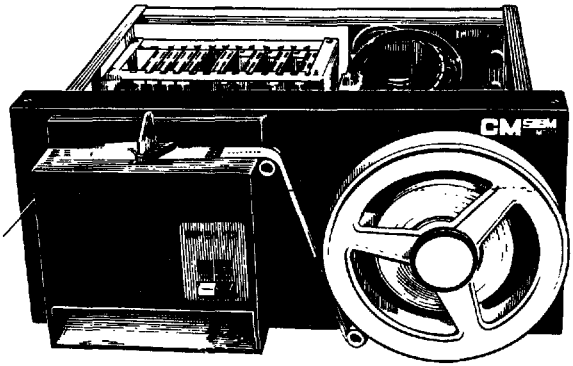


Рис. 7.6. Перфоратор СМ-6227

Один символ (или одно число не больше 255) кодируется на перфоленте набором пробивок (диаметром около 2 мм), расположенных в одной строке в поперечном направлении ленты. Чаще всего используется 8-дорожечная перфолента, хотя встречаются перфоленты с 5, 6 или 7 дорожками. В средней части перфоленты имеется непрерывная последовательность отверстий, предназначенных для перемещения перфоленты и (или) синхронизации считывания информации. Эти отверстия (диаметром около 1 мм) перфорируются на каждом шаге одновременно с перфорированием основных кодирующих отверстий. При кодировании в КОИ-7 восьмая дорожка может использоваться для размещения бита контроля четности (на рис. 7.5 единичное значение бита контроля четности показано пунктирной пробивкой).

При поступлении на перфоратор (рис. 7.6) кода символа с клавиатуры микроЭВМ или из ее памяти производится преобразование этого кода в набор сигналов для электромагнитов пуансонов, которые и пробивают заданные отверстия на ленте (если был получен код 0000 0000, то пробивается единственное отверстие на синхродорожке). После пробивки перфоратор (используя отверстие на синхродорожке) передвигает перфоленту примерно на 2,5 мм, подготавливая для перфорации следующую строку.

Скорость перфорации ограничивается как механическими свойствами ленты, так и конструктивными особенностями перфоратора и редко превышает 150 строк в 1 с.

Данные с перфоленты обычно вводятся фотосчитывающим устройством (рис. 7.7), состоящим из лентопротяжного механизма и блока считывания. Последний состоит из линейки фотосчитывающих элементов и осветителя, между которыми и протягивается перфолента. Когда при движении ленты свет начинает поступать на фотосчитывающий элемент, находящийся под синхродорожкой, производится считывание сигналов с остальных фотосчитывающих элементов, т. е. считывание кода строки. Такой способ считывания позволяет использовать непрерывное (а не стартстопное, как при перфорации) движение ленты, что увеличивает скорость считывания до 1500 строк в 1 с.

Печатающие устройства предназначены для получения информации из микроЭВМ в виде «твердой копии» — документа, который был бы удобен для прочтения человеком и мог храниться в течение нескольких лет.

Наиболее простое печатающее устройство — это электрическая рычажная пишущая машинка, снабженная схемой управления для связи с контроллером той или иной

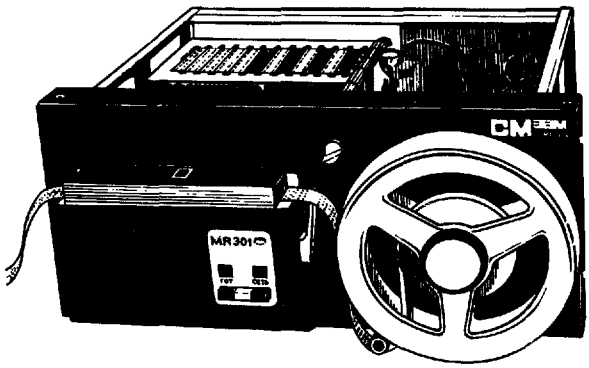


Рис. 7.7. Перфоленточное устройство ввода СМ-6203

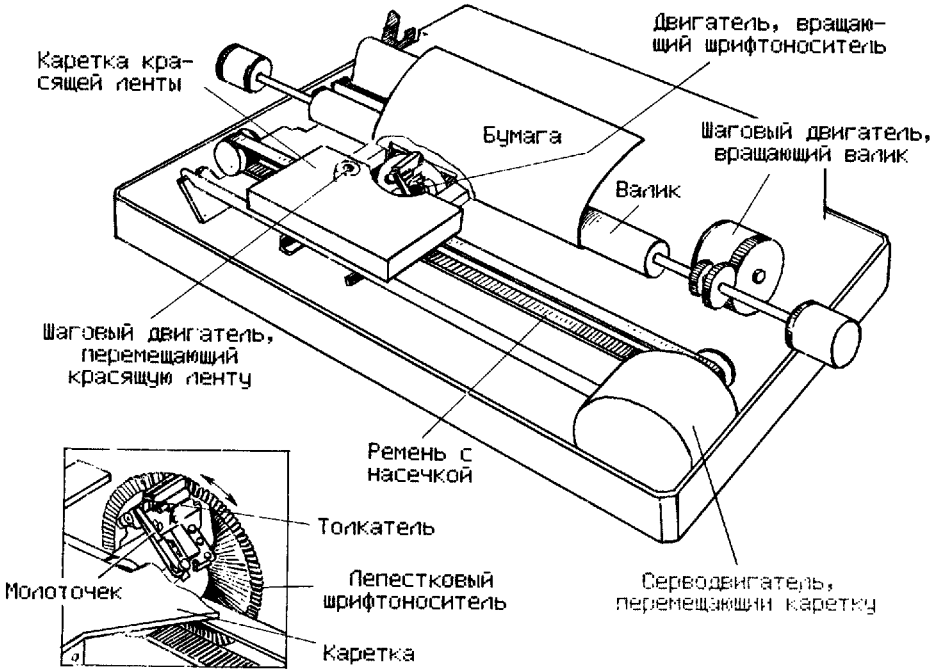


Рис. 7.8. Печатное устройство с лепестковым шрифтоносителем

микроЭВМ. Однако такие машинки имеют сравнительно невысокую скорость печати из-за больших затрат времени на перемещение механических устройств (рычага с заданной литерой и громоздкой каретки). Поэтому рычажные пишущие машинки с подвижной кареткой сейчас практически не используют в качестве внешних устройств микроЭВМ. Их заменяют на машинки с шрифтоносителем типа «ромашка» или матричным печатающим узлом.

Алфавитно-цифровое печатающее устройство (АЦПУ) типа «ромашка» (рис. 7.8) имеет неподвижную каретку, которая служит для закрепления бумаги и ее продвижения на новую строку. Вдоль каретки перемещается печатающий узел, в состав которого входят:

тонкий диск с лепестками, на которых расположены рельефные буквы, цифры и символы;

ударный рычаг с электромагнитом, который может прижать к бумаге (через красящую ленту) расположенный рядом с ним лепесток диска («ромашки»), т. е. отпечатать тот или иной знак;

электродвигатель, вращающий «ромашку» и подводящий к ударному рычагу лепесток, на котором находится нужный знак.

«Ромашка» может быть достаточно быстро и просто заменена на другую, что позволяет легко переходить на другие шрифт (например, курсив) и алфавит (например, латинский). Скорость печати АЦПУ типа «ромашка» около 50 знаков в 1 с.

Широко распространены матричные ударные АЦПУ, имеющие неподвижную каретку и подвижный печатающий узел, позволяющий воспроизвести нужный символ путем печати на бумаге ряда точек с помощью иголочек, приводимых в движение электромагнитами. Такой печатающий узел может содержать, например, 35 иголочек для одновременной печати всех точек символа, синтезируемого с помощью матрицы 5×7 (скорость печати подобных АЦПУ достигает 200 зн./с).

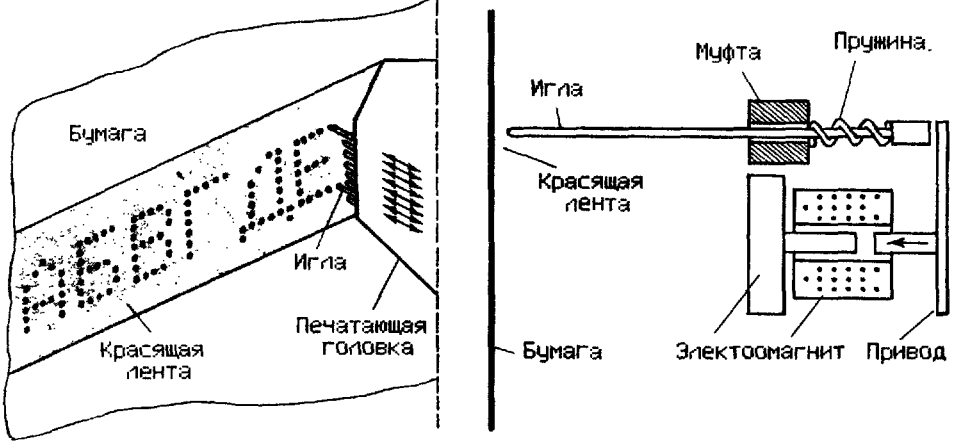


Рис. 7.9. Принцип действия матричного печатающего устройства

Однако в дешевых (и менее быстродействующих) АЦПУ матричного типа используется один вертикальный ряд иголочек и все точки синтезируемого символа появляются на бумаге лишь после перемещения печатающего узла на несколько (например, пять) шагов. На рис. 7.9 показан принцип действия матричного АЦПУ (8-я игла используется для печати нижних выносных элементов у букв Д, Ц и т. п.), а на рис. 7.10 — внешний вид одного из таких устройств.

При синтезе символов из большего числа точек появляется возможность воспроизводить символы разных шрифтов и алфавитов, а также графическую информацию (рис. 7.11).

Наиболее дешевые АЦПУ — термографические печатающие устройства. Примером такого устройства с построчной печатью символов, синтезированных из точек, является линейка маленьких нагревательных элементов, через которую протягивается специальная бумага. Если на какой-либо нагревательный элемент подан импульсный электрический сигнал, то такой элемент кратковременно нагревается и на прижатой к нему термочувствительной бумаге появляется темная точка. Это позволяет «напечатать» один горизонтальный ряд одновременно во всех матрицах, синтезирующих символы целой строки. После перемещения бумаги «печатается» следующий ряд и т. д., пока не будут воспроизведены все символы строки.

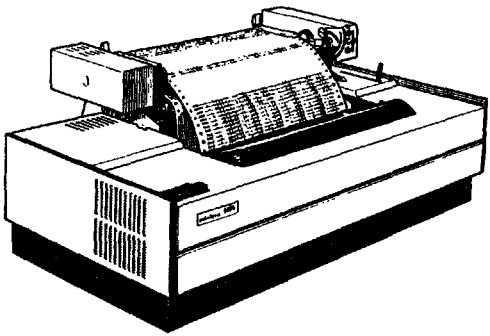


Рис. 7.10. Алфавитно-цифровое печатающее устройство последовательного действия «Роботрон-1154»

Скорость печати таких устройств доведена до 30 строк в 1 с. Однако термографические устройства не находят широкого распространения из-за необходимости использования специальной бумаги, выпуск которой ограничен. Кроме того, изображение на термочувствительной бумаге со временем тускнеет.

К перспективным безударным АЦПУ относится струйная печать: наэлектризованные мельчайшие капельки чернил направляются на бумагу электрическим (магнитным) полем, которое может отклонять поток капелек или прерывать его, формируя

Рис. 7.11. Примеры печати шрифтов и графической информации на АЦПУ «Роботрон-1154»

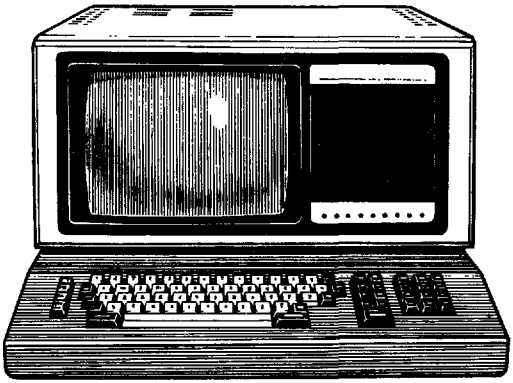


Рис. 7.12. Дисплей СМ-7301, предназначенный для вывода алфавитно-цифровой (24 строки длиной 80 символов) и графической (512×256 точек) информации

воспроизводимый символ. Бесшумные струйные АЦПУ обладают высокой скоростью печати (до 100 знаков в 1 с), хорошей разрешающей способностью (до 1000 точек в знаке), возможностью печати на различных материалах (бумаге, ткани и т. д.) и многоцветной печати (до 1000 различных цветов и оттенков).

Дисплей (от англ. display – показывать, визуализировать) в строгом смысле этого слова представляет собой экран любого типа (индикаторную панель, жидкокристаллический экран и т. п.) и электронную схему, преобразующую цифровые коды (поступающие из ЭВМ) в изображение символов и линий на экране. Однако чаще всего дисплеем называют устройство с электронно-лучевой трубкой (индикаторный дисплей) и даже индикаторный дисплей, дополненный клавиатурой (рис. 7.12) и световым пером для ручного ввода и модификации пользователем отображаемых данных (диалоговый дисплей).

В дисплее (как и в телевизоре) электронный луч перемещается по экрану слева направо. Яркость свечения экрана определяется энергией электронного луча; управляя этой энергией, можно добиваться разной яркости различных участков горизонтальной строки. Во время обратного хода (справа налево) электронный луч гасится и поэтому не оставляет на экране заметного следа. Затем луч начинает снова равномерно двигаться слева направо, но уже в следующей строке, и такой процесс растровой развертки продолжается до завершения сканирования всего экрана. Полная развертка растра выполняется 50 раз в секунду. Каждая строка развертки образуется несколькими сотнями дискретных точек (рис. 7.13, *а*), называемых элементами изображения (ЭИ) или пикселами (пиксел — от picture element).

Дисплеи с высокой и низкой разрешающей способностью различаются по числу адресуемых машиной пикселов: обычно низкой разрешающей способности соответствуют $128 \times 48 = 6144$ пикселов, а высокой — 1000×1000 пикселов. В дисплеях с высокой разрешающей способностью необходимо в течение пятидесятой доли секунды передавать на электронно-лучевую трубку более миллиона битов информации, что вызывает большой непроизводительный расход машинных ресурсов, избежать которого можно несколькими способами. Один из них — уменьшение частоты регенерации изображения (смены кадра), но такой способ приводит к утомительному для глаза мерцанию экрана. Второй способ — уменьшение числа строк развертки и дискретных точек растра, но при этом формируется довольно крупнозернистое изображение.

Возможно также использование специальной запоминающей электронно-лучевой трубки, которая способна длительно сохранять заданный уровень яркости свечения. Однако такие трубки стоят дороже, и их применение связано с большими непроизводительными затратами машинных ресурсов, когда изображение на экране должно быть заменено новыми. Наконец, еще один способ — применение дисплея с произвольной разверткой электронного луча по экрану (рис. 7.13, *б*). Такие полноадресуемые дисплеи обеспечивают важные преимущества при необходимости вычерчивания линий и векторного представления изображений.

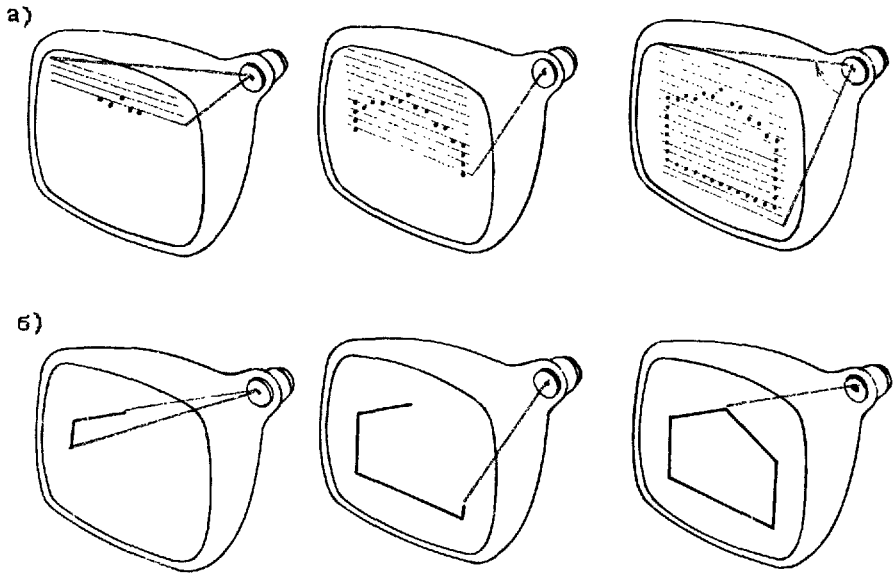


Рис. 7.13. Два режима построения изображения на экране дисплея:
а — растровый; *б* — векторный

Обычный телевизионный экран, хотя и недорог, не удовлетворяет предъявляемым требованиям, поскольку не способен давать 80 знаков в строке, как того требует технология обработки текстовой информации, при одновременном обеспечении разрешающей способности, приемлемой для удобства зрительного восприятия изображения человеком. Во многих случаях ощутимые преимущества можно получить от цветных изображений вместо монохроматических (обычно черно-белых или черно-зеленых) с эквивалентной разрешающей способностью.

Современные дисплеи широкого применения выполнены на одноцветных (реже многоцветных) электронно-лучевых трубках с диагональю 30—50 см. Они, как правило, позволяют одновременно воспроизводить как алфавитно-цифровую информацию (25 строк по 80 знаков в каждой), так и графическую информацию с достаточно высокой разрешающей способностью (640×200 элементов изображения). Кроме традиционных возможностей по редактированию текста на экране (вставка или изъятие любой строки, любого символа в строке или группы символов) многие из этих дисплеев могут:

выделять отдельные символы или участки текста путем изменения интенсивности их свечения, мерцания этих символов или вывода в позитиве (черные символы на белом фоне);

выделять на экране различные области (окна), размещать в этих окнах различные тексты (чертежи) и переходить к работе с нужным (активным) окном, используя при необходимости текст, расположенный в других окнах (например, создавая новый текст из нескольких старых); на одноцветных дисплеях выделение производится различной интенсивностью свечения окон, а на цветных — различным цветом окон;

сохранять на магнитных дисках те или иные области экрана с последующим их воспроизведением в нужный момент;

выводить изображения задаваемых графических объектов (линий, прямоугольников, эллипсов и других фигур), а также совмещать фрагменты текста и графики.

позволяют ускорить и упростить для человека ввод информации в ЭВМ и получать «подсказки» от ЭВМ при выполнении какой-либо работы, не отвлекаясь при этом на просмотр документов или экрана дисплея.

Речь представляет собой последовательность дискретных звуковых сегментов, называемых фонемами, которые связаны друг с другом во времени. Каждой фонеме свойственны собственные различимые звуковые характеристики. Произносимое слово состоит из нескольких слогов, каждый из которых образуется несколькими фонемами; характеристики любой данной фонемы зависят от ее непосредственного фонетического окружения, или условий сочленения.

Последнее обстоятельство приводит к тому, что одна и та же фонема имеет различное звучание в разных словах, а это затрудняет задачу разделения конкретного слова на составляющие фонемы. Кроме того, поскольку эффект сочленения может распространяться и за пределы слов, распознавание связной речи гораздо труднее распознавания отдельных слов. Задача еще больше усложняется тем, что произношение у разных людей различно и даже не всегда одинаково у одного и того же человека.

Упрощения устройств (программ) распознавания речи добиваются путем ограничения их словаря (до нескольких десятков слов) и настройки системы на конкретного «диктора». Перед началом применения системы пользователь формирует определенный словарь. Затем он несколько раз произносит каждое слово, чтобы дать возможность машине запомнить варианты произношения, и после этого система оказывается готовой к работе. Она «слушает» слова, произносимые пользователем, и сопоставляет их с эталонами, хранящимися в словарной базе данных. Если обнаруживается совпадение, идентифицируется соответствующее слово; в противном случае пользователь делает повторную попытку речевого ввода либо вводит информацию традиционным способом (например, с клавиатуры).

Синтез речи, или процесс формирования речевых сигналов из фрагментов цифровых данных, находящихся в памяти ЭВМ, проще всего реализуется путем составления ответа из слов, введенных пользователем на этапе обучения ЭВМ. В этом случае машина выводит информацию голосом пользователя.

Ведутся большие работы по «обучению» ЭВМ естественному языку, и по мере увеличения мощности и снижения стоимости и габарита микроЭВМ речевой обмен информацией с ней станет таким же обычным, как и обмен с помощью диалогового дисплея.

7.3. Устройства для связи микроЭВМ с объектами управления

МикроЭВМ управляют станками и игровыми автоматами, ходом научных экспериментов и доильными аппаратами, автомобилями и роботами. В этих (а также других) устройствах, приборах и системах используется много серийных электрических датчиков и исполнительных органов (например, электродвигателей), работающих с непрерывными (аналоговыми) электрическими сигналами (см. параграф 1.1). Для связи таких датчиков и исполнительных органов с микроЭВМ требуется преобразовать аналоговый сигнал в число, пропорциональное амплитуде этого сигнала, и наоборот.

Процедура аналогоцифрового преобразования состоит из двух этапов: дискретизации по времени (выборки) и квантования по уровню. Процесс дискретизации состоит из измерения значений непрерывного сигнала $X(t)$ только в дискретные моменты времени 0, T, $2T$, ..., отстоящие друг от друга на величину периода дискретизации T (рис. 7.14).

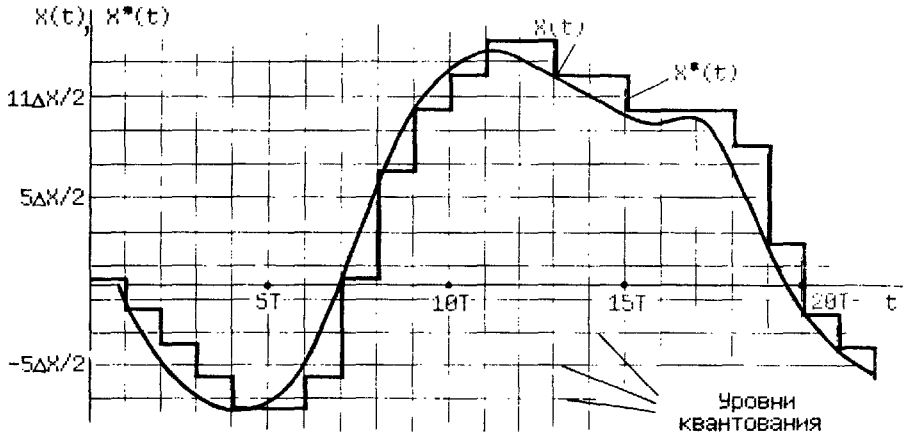


Рис. 7.14. Дискретизация и квантование непрерывного сигнала $X(t)$

Такой измерительный процесс, если он реализуется без необходимых мер предосторожности, может привести к возникновению значительных погрешностей. Важно иметь достаточное число отсчетов за единицу времени для правильного представления существенной информации, содержащейся в высокочастотной составляющей этого сигнала. В то же время частые измерения требуют увеличения быстродействия преобразования и обработки, а следовательно, сложности и стоимости преобразователя. Поэтому темп дискретизации необходимо поддерживать на минимально допустимом уровне, при котором составляющая общей погрешности, обусловленная процессом дискретизации, не превышает установленных пределов.

Для квантования диапазон изменения входного сигнала подразделяется на равные интервалы (уровни квантования): ..., $-5\Delta X/2$, $-3\Delta X/2$, $-\Delta X/2$, $\Delta X/2$, $3\Delta X/2$, $5\Delta X/2$, ..., где ΔX — интервал (шаг квантования). Операция квантования сводится к определению того интервала, в который попало дискретизированное значение $X(t)$, и к присваиванию выходному значению $X^*(t)$ цифрового кода, соответствующего значению центра найденного интервала. При такой замене может быть допущена ошибка, равная $\Delta X/2$. Для ускорения процесса преобразования, упрощения и удешевления преобразователя надо выбирать максимально допустимый шаг квантования, при котором еще не появляются большие погрешности.

При проектировании систем, в которых должны использоваться аналого-цифровые или цифроаналоговые преобразователи, следует иметь хотя бы минимальное представление о принципах работы этих преобразователей. В противном случае вероятность правильного выбора подходящего серийного преобразователя будет близка к нулю. Знакомство же с такими преобразователями лучше всего начинать с цифроаналоговых преобразователей.

Цифроаналоговый преобразователь (ЦАП) служит для преобразования числа в пропорциональную аналоговую величину, представленную в виде электрического напряжения, тока и т. п. Цифровая информация вводится чаще всего в виде параллельного кода преобразуемого числа, а аналоговая информация представлена в виде одного сигнала, величина которого является носителем информации.

Существует множество способов построения ЦАП, но наиболее часто применяют схемы, построенные на операционном усилителе и резисторах (рис. 7.15). Усилитель, работающий в режиме сумматора, имеет n входных

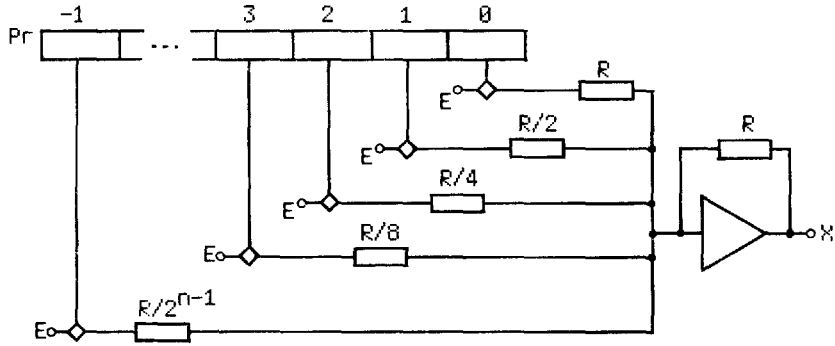


Рис. 7.15. Цифроаналоговый преобразователь с весовыми резисторами

резисторов, где n — число двоичных разрядов, используемых для представления преобразуемого числа X^*. Сопротивления этих резисторов — $R/2^0$, $R/2^1$, ..., $R/2^{n-1}$.

Один зажим каждого из резисторов подсоединен к электронному ключу, состояние которого определяется значением соответствующего разряда преобразуемого числа, находящегося в регистре *Рг*. Если двоичный разряд — единица, то резистор соединяется с источником опорного напряжения E; в противном случае ключ разомкнут. Так как величина выходного сигнала такой схемы равна отрицательной сумме входных напряжений с весовыми коэффициентами, равными отношению сопротивления обратной связи усилителя к соответствующему входному сопротивлению, то при $X^*=0\ldots001$ сигнал $X=-E$, при $X^*=0\ldots010-X=-2E$, при $X^*=0\ldots011-X=-3E$ и т. д.

Для получения сигнала со знаком преобразуемое число представляют в дополнительном коде и его старший разряд (знаковый разряд) используют для управления ключом, который может добавить к X напряжение $+2^{n-1}E$. Так, если $X^*=11\ldots111$ (-1 в дополнительном коде), то $X=-E(2^{n-2}+2^{n-3}+\ldots+2+1)$ и после суммирования этого сигнала с $+2^{n-1}E$ выходное напряжение ЦАП будет равно $+E$, т. е. пропорционально -1.

Существует модификация рассмотренного ЦАП, в которой для уменьшения количества номиналов резисторов используют цепную схему, составленную из вдвое большего числа резисторов всего лишь двух номиналов — R и $2R$. Есть также и ЦАП, основанные на других принципах преобразования цифрового кода в напряжение или ток.

Нетрудно заметить, что для повышения точности представления числа недостаточно простого увеличения разрядности ЦАП. Необходимо также повышать точность изготовления резисторов, стабильность опорного напряжения E, линейность операционного усилителя и ряд других параметров схемы. Поэтому выбор того или иного серийного ЦАП только по одному параметру — разрядности является грубейшей ошибкой.

Аналогоцифровой преобразователь (АЦП) служит для преобразования аналогового сигнала (например, электрического напряжения) в код числа. Существует много способов построения АЦП, каждый из которых обладает своими достоинствами и недостатками. В серийных полупроводниковых интегральных АЦП наибольшее распространение получили два из них: последовательное приближение и параллельное преобразование.

В АЦП, построенных по методу последовательного приближения, используются регистр последовательного приближения, ЦАП, сравнивающее

устройство (компаратор), тактовый генератор и схема управления. Перед измерением все разряды регистра устанавливаются в 0. При получении приказа на преобразование каждый из разрядов регистра, начиная со старшего, поочередно устанавливаются в 1.

Если выходной сигнал ЦАП, связанного с этим регистром, не превышает уровня входного аналогового сигнала, разряд остается в состоянии 1; в противном случае разряд сбрасывается обратно в 0. Для n-разрядного АЦП необходимо совершить n таких шагов. Подобный процесс может быть представлен как двоичный поиск, начинающийся с середины. В модуле АЦП последовательного приближения имеются вход «Начало преобразования» и выход «Преобразование выполнено». Все такие преобразователи имеют параллельный цифровой выход (все разряды выводятся одновременно по n-разрядной шине) и, как правило, еще и последовательный выход (n разрядов выходного кода выдаются последовательно, начиная со старших разрядов, по одной линии).

АЦП последовательного приближения обладают относительно высокой точностью и достаточно высоким быстродействием: для n-разрядного преобразования требуется выполнить только n обращений к ЦАП, который может выполнить одно преобразование за время от десятых долей до нескольких микросекунд (в зависимости от типа и разрядности). Для того чтобы изменение аналогового сигнала во время от начала преобразования до его окончания не влияло на точность преобразования, обычно используют специальные схемы, запоминающие аналоговый сигнал. Устройства выборки-хранения обеспечивают выборку мгновенного значения аналогового сигнала и хранение его с заданной точностью во время преобразования в АЦП. Существуют АЦП и со встроенными схемами выборки-хранения.

Значительно бóльшим быстродействием отличаются АЦП параллельного кодирования. В них для преобразования аналогового сигнала в n-разрядный двоичный код используется 2^{n-1} компараторов (сравнивающих устройств). На один из двух дифференциальных входов каждого компаратора подается свое опорное напряжение, формируемое резисторным делителем. Разность между опорными напряжениями двух ближайших компараторов равна $E/2$, где E — опорное напряжение, соответствующее максимальному значению преобразуемого аналогового сигнала. Другие входы компараторов объединены, и на них подается входной сигнал. Приоритетный шифратор формирует выходной цифровой сигнал, соответствующий самому старшему сработавшему компаратору.

Способ параллельного кодирования (иногда он называется способом мгновенного кодирования) отличается наибольшим быстродействием. Время задержки при передаче сигнала от входа к выходу равно сумме запаздываний компараторов и шифратора и составляет несколько десятков наносекунд. Параллельные преобразователи, выпускаемые промышленностью, имеют от 16 до 256 уровней квантования (от 4 до 8 разрядов выходного кода). При большем числе разрядов АЦП становятся чрезмерно дорогостоящими и громоздкими.

В некритичных к быстродействию применениях и при желании получить высокую точность при сравнительно низкой стоимости и помехоустойчивости АЦП используют преобразователи, построенные по методу двойного интегрирования. Такие преобразователи состоят из конденсатора (интегрирующий элемент), который в течение фиксированного промежутка времени заряжается током, точно пропорциональным входному сигналу, после чего он разряжается постоянным током до тех пор, пока напряжение на нем не станет равным нулю. Время разряда конденсатора пропорционально значению входного сигнала и используется для подсчета тактовых импульсов

фиксированной частоты при помощи счетчика. Полученное число, пропорциональное входному уровню, является выходным цифровым сигналом.

Способ двойного интегрирования позволяет получить высокую точность, не предъявляя чрезмерно высоких требований к стабильности компонентов. В частности, здесь нет необходимости иметь высокостабильный конденсатор, поскольку скорость изменения его напряжения как на стадии заряда, так и разряда обратно пропорциональна емкости *C*. Более того, не предъявляются высокие требования даже к стабильности частоты. Дело в том, что на первой стадии фиксированный интервал интегрирования получают делением частоты тех же самых тактовых импульсов, которые потом используются при счете. Если тактовая частота снизится на 10 %, то на 10 % повысится уровень, которого достигнет линейно нарастающее напряжение на первой стадии, и соответственно на 10 % увеличится время разряда. Так как последнее отсчитывается с помощью тех же тактовых импульсов, частота которых снизилась на 10 %, то окончательное число получится неизменным.

Существуют и другие типы АЦП, обладающие теми или иными достоинствами в конкретных приложениях.

Все серийные интегральные АЦП снабжены схемами управления, позволяющими легко использовать эти преобразователи в системах микропроцессорной техники (входы и выходы «Запуск», «Цикл», «Конец преобразования» и т. п.). Кроме того, в состав АЦП могут входить (или включаться) многоканальные аналоговые коммутаторы, позволяющие применять АЦП для обработки сигналов с 4, 8, 16 или 32 датчиков с последовательной или произвольной выборкой коммутируемого канала. Подобные схемы служат и для коммутации выходов ЦАП.

7.4. Внешние запоминающие устройства большой емкости

Вспомогательные запоминающие устройства обеспечивают большой объем памяти и относительно недороги, но обладают меньшим быстродействием, чем основная память микроЭВМ. Почти все такие ЗУ предназначены для долговременного хранения программ и данных и снабжены механизмом для установки и съема носителей информации, что дает возможность физически переносить программы и данные с одной машины на другую.

В настоящее время наиболее широкое распространение получили внешние запоминающие устройства на магнитных носителях (лентах и дисках, покрытых тонким слоем магнитного материала).

Принцип магнитной записи цифровой информации поясняет рис. 7.16, *а*. Основу процесса магнитной записи составляет взаимодействие магнитного носителя информации (запоминающей среды) и магнитных головок: миниатюрных электромагнитов, располагаемых у поверхности движущегося магнитного носителя с небольшим зазором при бесконтактной записи или без зазора при контактной записи. Информация записывается на носителе вдоль дорожки, проходящей под головкой, путем подачи на головку тока записи, который создает магнитный поток, проходящий через зазор головки и частично через магнитный слой носителя. Изменением направления сигнала в обмотке головки изменяют полярность намагничивания. Считывание информации производится при прохождении под головкой дорожки носителя с записанной информацией. Перемещение под зазором границы между участками с разной полярностью намагничивания создает в обмотке импульс той или иной полярности.

Имеется несколько способов магнитной записи цифровой информации. При поляризованной записи с возвращением к нулю 1 и 0 формируются на

Рис. 7.16. Магнитная запись цифровой информации:

а — принцип записи; *б* — поляризованная запись с возвратом к нулю; *в* — запись с возвратом к уровню предварительной намагниченности; *г* — запись без возврата к нулю с переключением потока по единицам; *д* — запись с частотной модуляцией; *е* — запись с модифицированной фазовой модуляцией

размагниченном носителе током разной полярности (рис. 7.16, *б*). Достоинство этого способа — отсутствие необходимости записи синхроимпульсов или стабилизации и выравнивания скоростей записи и считывания, так как любое значение записанного бита (как 0, так и 1) представлено определенным уровнем и полярностью намагничивания запоминающей среды. Недостатки способа — необходимость размагничивания носителя перед записью и сравнительно малая амплитуда сигнала при считывании (вдвое меньшая, чем при других способах записи).

На рис. 7.16, *в* представлен способ записи на носитель, предварительно намагниченный до уровня насыщения, который обычно соответствует значению 0. Поэтому при записи здесь отсутствуют сигналы, соответствующие 0, и ток записи всегда имеет одно направление, создавая намагниченность до уровня насыщения, соответствующего 1. Недостаток способа записи с возвращением к уровню предварительной намагниченности заключается в отсутствии самосинхронизации, поскольку нули в коде не порождают импульсов тока при считывании.

Следовательно, такой способ можно использовать в тех накопителях, где одновременно считывается информация с нескольких дорожек, одна из которых синхронизирующая.

Несколько странным выглядит способ записи без возврата к нулю с переключением потока по единицам (рис. 7.16, *г*). Здесь, как и в предыдущем способе, никак не отмечается состояние, соответствующее 0, зато 1 отмечается изменением направления намагничивания, а не полярностью намагничивания, что повышает надежность считывания. Для записи не требуется предварительного намагничивания носителя или стирания ранее записанной информации. Однако и в этом способе отсутствует самосинхронизация и он обычно используется для записи информации на многодорожечную

магнитную ленту, где имеется контрольная дорожка, обеспечивающая синхронизацию при считывании.

Самосинхронизирующимся способом, которому присущи все достоинства предыдущего способа, является запись с частотной модуляцией (рис. 7.16, *д*). В нем ток записи изменяет направление на границе каждого такта записи (каждого бита) и, кроме того, посредине такта при записи 1. Следовательно, при записи единиц частота переключения тока вдвое больше, чем при записи нулей, что дало второе название способу — двухчастотное кодирование.

Наконец, отметим трехчастотный способ записи, или запись с модифицированной фазовой модуляцией (рис. 7.8, *е*). Переключение тока в магнитной головке происходит при записи 1 в начале такта записи, а при записи 0 — посредине такта, если только следующий записываемый бит не 1. В противном случае при записи 0 переключения тока не производится. Этот самосинхронизирующийся способ позволяет получить бо́льшую плотность записи, чем другие методы.

Накопители на магнитных дисках — наиболее распространенный тип внешних запоминающих устройств вычислительных машин, а в микроЭВМ широко используются накопители на гибких магнитных дисках, часто называемых дискетами.

Дискета состоит из конверта (или другого защитного чехла), в котором расположен гибкий диск, похожий на гибкую грампластинку, выпускаемую, например, фирмой «Мелодия» (рис. 7.17). Однако этот диск гладкий (без спиральной звуковой канавки) и покрыт износоустойчивым ферролаком. Так как в момент записи электромагнитная головка может располагаться над вращающимся диском лишь в строго фиксированных позициях, определяемых механизмом ее перемещения вдоль радиуса диска, то информация записывается на ряд концентрических дорожек. Такие дорожки не видны на диске — это просто та часть его поверхности, микроскопические участки которой намагничены в разных направлениях, воспринимаемых при чтении в виде нулей и единиц. Дорожка разделена на ряд секторов, и, как правило, за один цикл записи или считывания обращение производится только к одному сектору.

В зависимости от установленного формата одна дорожка имеет от 8 до 26 секторов, а в каждый сектор записываются 128 512 байт данных.

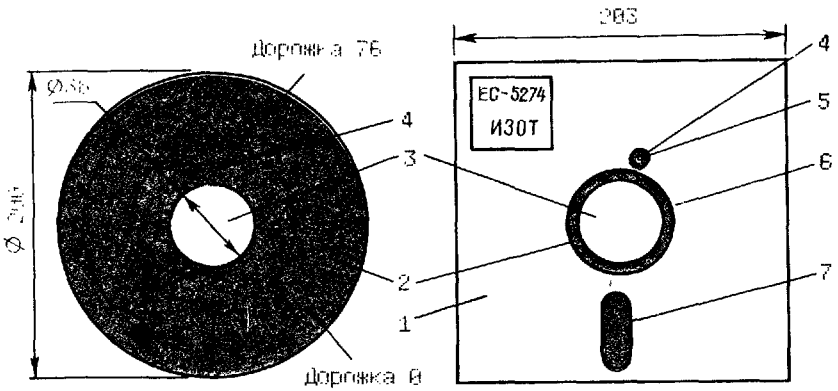


Рис. 7.17. Дискета ЕС-5274:

1 — конверт; *2* — гибкий диск; *3* — отверстия для привода диска; *4* — индексные отверстия; *5* — прорезь для доступа к индексному отверстию; *6* — прорезь для доступа к приводному отверстию; *7* — прорезь для подвода к диску подвижной электромагнитной головки

Полная емкость памяти на гибком диске варьируется в зависимости от плотности записи данных на одной дорожке (продольная плотность достигает 275 бит/мм), плотности размещения концентрических дорожек (поперечная плотность — до 60 дорожек на 1 мм по радиусу диска) и числа секторов, на которые разбита каждая дорожка. Емкость большинства современных гибких дисков равна 125—500 Кбайт.

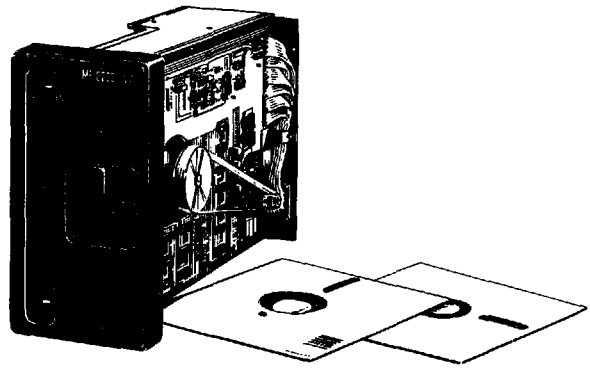


Рис. 7.18. Накопитель на гибких магнитных дисках

На дискете ЕС 5274 информация представлена в последовательном коде на 77 концентрических дорожках. Каждая дорожка разделена на 26 секторов, а каждый сектор содержит служебную информацию (включающую адрес сектора) и 128 байт данных (в некоторых ЭВМ, например в персональной ЭВМ «Искра 226», дорожка разбивается на 13 секторов, имеющих по 256 байт данных, т. е. аналогичный объем информации). Таким образом, на одной стороне дискеты могут храниться $77 \times 26 \times 128 = 256\,256$ байт информации, а на двух сторонах (обратная сторона дискеты схожа с лицевой) — более чем 0,5 Мбайт информации, т. е. примерно половина текста данной книги.

Для записи и считывания дискет используются накопители на гибких магнитных дисках (НГМД). Один из таких накопителей, встраиваемых в различные корпуса, показан на рис. 7.18. Чаще всего в отдельный корпус или корпус персональной ЭВМ встраивают два НГМД, что не только позволяет удвоить объем доступной для ЭВМ информации, но и облегчает перезапись информации с одной дискеты на другую.

Включив накопитель и открыв его дверцу (рис. 7.19), пользователь вставляет дискету в щель и перемещает ее по направляющим до упора, обеспе-

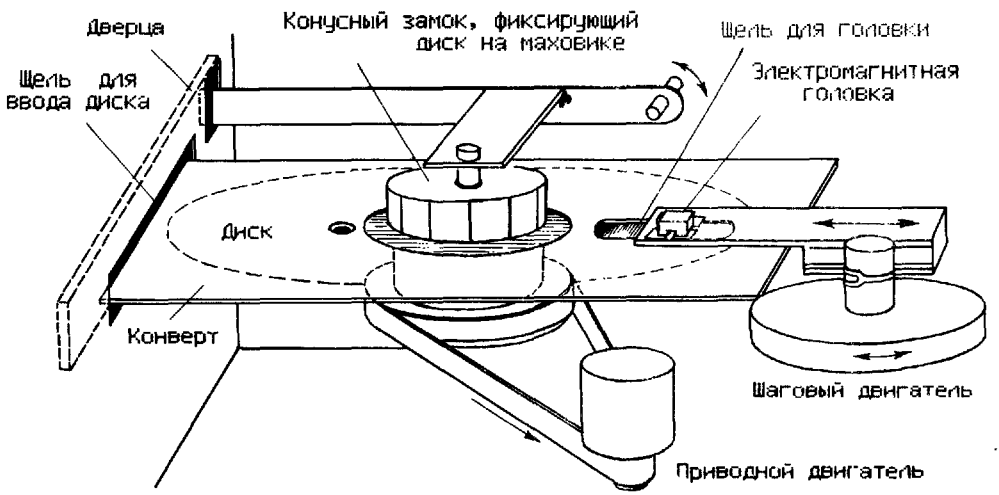


Рис. 7.19. Конструкция НГМД (отсутствуют фотодатчик с осветителем и электромагнит, поджимающий диск к головке)

чивая размещение приводного отверстия (поз. *3* на рис. 7.17) над конусным выступом вращающегося маховика. При закрывании дверцы приходит в действие прижимная система, которая автоматически центрирует диск и прижимает его к выступу маховика. Диск начинает вращаться с частотой 360 об/мин прямо в конверте, внутренняя поверхность которого обклеена мягким материалом, обеспечивающим легкое вращение диска и его очистку от пыли.

В процессе вращения диска его индексное отверстие периодически проходит между осветителем и фотодатчиком, который информирует контроллер НГМД о начале дорожки (номер этой дорожки определяется положением электромагнитной головки). Головка может вести как запись, так и чтение информации. Она перемещается над специальной прорезью в конверте с помощью шагового двигателя с ленточным (а иногда винтовым) приводом. Один шаг вращения этого двигателя соответствует линейному смещению головки на одну дорожку.

Информация о необходимости записи или считывания содержимого заданного сектора какой-либо дорожки пересылается из микроЭВМ в контроллер НГМД, который подает сигналы на шаговый двигатель для перемещения головки (во время этой операции головка не соприкасается с диском). После того как головка установилась над заданной дорожкой, контроллер ждет от индексного отверстия сигнала о начале дорожки, а затем, сообразуясь с частотой вращения диска и длиной сектора, пропускает все секторы, предшествующие заданному.

Наконец, контроллер подает сигнал на электромагнит, поджимающий диск к электромагнитной головке, и начинает управлять записью или считыванием информации. Естественно, что последняя операция приводит к износу как диска, так и электромагнитной головки. Однако современные НГМД позволяют (при соответствующем уходе) выполнить не менее миллиона записей (считываний) на (с) каждой из дорожек гибкого диска. (Ранее утверждалось, что дорожки не видны, однако износ поверхности диска постепенно проявляет эти дорожки.)

В заключение отметим, что стоимость гибкой дискеты соизмерима со стоимостью качественных кассет кассетных магнитофонов.

Выше уже упоминалось о физическом размещении информации на дискете. На рис. 7.20 представлен один из возможных вариантов такого размещения, который получается после выполнения специальной операции, называемой форматированием диска. При форматировании контроллер устанавливает головку на дорожку с номером 0 и (после получения сигнала о появлении импульсного отверстия) начинает посылать в обмотку головки токовые сигналы, осуществляя запись информации определенного вида.

Сначала записывается байт со специальным кодом, который при дальнейшей работе с диском будет идентифицироваться как маркер начала дорожки. Затем записываются 32 нулевых байта и далее 26 секторов по 188 байт в каждом. При такой записи на дорожке за 26 сектором остается промежуток, на котором можно было бы записать еще 320 байт информации (домаркерная зона). Этот промежуток нужен для компенсации изменения скорости вращения диска (при увеличении скорости промежуток будет уменьшаться). Кроме того, этот и другие промежутки, показанные на рис. 7.20, позволяют выделить временные интервалы, необходимые для выполнения тех или иных операций при записи (считывании) информации.

Информация, записываемая в каждый из секторов, ясна из рис. 7.20. Следует лишь отметить, что в поле данных при форматировании записываются нули, что позволяет затем произвести контроль правильности записи и считывания информации, т. е. контроль качества дискеты.

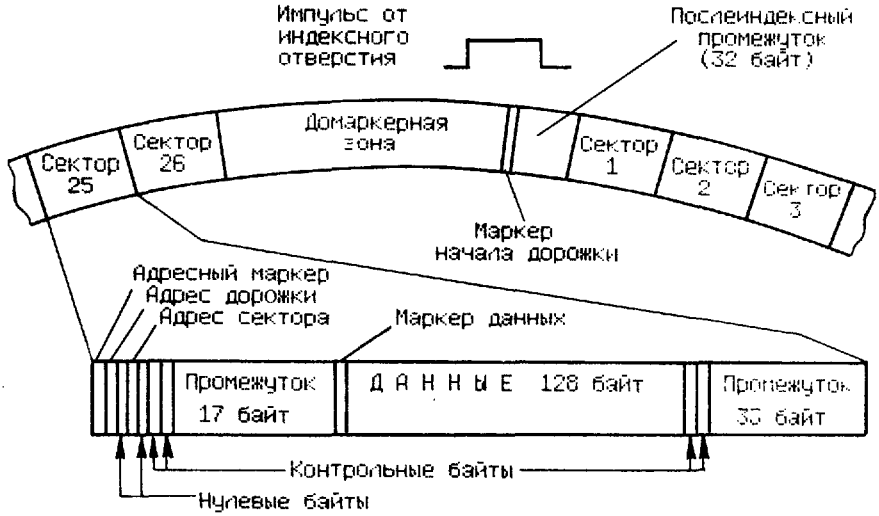


Рис. 7.20. Формат дискеты (один из вариантов)

После окончания форматирования дорожки с номером 0 контроллер перемещает магнитную головку на дорожку с номером 1 и повторяет все операции, которые были описаны выше. Так последовательно форматируются все 77 дорожек дискеты.

В рассмотренном программном разделении на секторы (мягком секторировании) полезная информация составляет менее 70 % общего числа записанных байтов. Это привело к появлению дискет с жестким секторированием: в дополнение к индексному отверстию (начало дорожки) на диске сделаны еще 32 равномерно расположенных отверстия (начало 32-х секторов). При тех же 128 байт данных в каждом из секторов общий объем полезной информации на дискете увеличивается в 32/26 = 1,22 раза (за счет сокращения промежутков).

В некоторых НГМД устанавливают блоки из двух магнитных головок, между которыми при записи (считывании) зажимается гибкий диск. И хотя в каждый конкретный момент времени токовые посылки поступают только в одну из этих головок, двусторонний диск фактически удваивает емкость ЗУ. Совершенствование технологии изготовления НГМД и дискет позволило начать выпуск НГМД с вдвое большей плотностью записи (за счет удвоения числа дорожек). Это привело к созданию нового поколения НГМД с двусторонней записью и двойной плотностью. Общая емкость таких дискет доведена до 3 Мбайт.

Кроме дискет диаметром 200 мм выпускают мини-дискеты диаметром 130 мм. Они также подразделяются на дискеты с мягким и жестким секторированием. Несмотря на меньшие размеры, общий объем информации, сохраняемой на мини-дискете, немногим отличается от аналогичного объема информации, сохраняемой на дискете диаметром 200 мм. В последнее время стали появляться НГМД с микродискетами диаметром 89 мм. Их емкость доведена до 0,72 Мбайт.

Основная характеристика производительности НГМД — время обработки одного обращения по чтению или записи. Это время для разных типов НГМД изменяется в пределах от 200 до 500 мс. В случае одиночного обращения за данными такое время обработки можно считать достаточно малым, но в некоторых видах обработки (например, обработка текстовой инфор-

мации) отдельные команды программы могут потребовать до 20 и более обращений к НГМД, на что может расходоваться 80–90 % общего времени выполнения команд. Поэтому там, где необходима более высокая производительность, применяют не гибкие, а жесткие диски, время обращения к которым на порядок меньше.

Имеется множество типов накопителей с жесткими магнитными дисками (НМД). В них используются алюминиевые диски с ферролаковым покрытием или пакеты из 2–12 таких дисков. Количество головок, которые могут одновременно записывать (считывать) информацию, обычно равно $2n-2$, где n – количество дисков в пакете (не используются лишь наружные поверхности крайних дисков пакета). Головки поджимаются к быстровращающимся дискам (2000—2400 об/мин) и благодаря аэродинамическому эффекту отделены от дисков тонкой пленкой уплотненного воздуха (толщиной около 4 мкм). Все эти факторы позволяют ускорить обмен информацией между ЭВМ и НМД. Емкость таких ЗУ достигает 100 Мбайт.

Однако рассмотренные выше НМД достаточно громоздки и дороги, что определяет основную область их применения: большие ЭВМ и мини-ЭВМ. В микроЭВМ стали применять так называемые винчестерские диски – накопители на жестких дисках с подвижной головкой, расположенные в герметизированном корпусе. Благодаря такой конструкции удается более эффективно препятствовать проникновению внутрь пыли и других загрязняющих частиц из окружающей среды и, следовательно, помещать магнитные головки в десять раз ближе к поверхности диска, чем в других типах дисковых ЗУ. Отсюда появляется возможность уплотнения записи информации и сокращения времени обращения к ней.

Винчестерские диски с диаметром 133 и 200 мм имеют (в зависимости от числа дисков в пакете) емкость от 3 до 300 Мбайт и сравнительно низкую стоимость (по сравнению с НМД аналогичной емкости и производительности). Однако в отличие от гибкого диска и НМД со съемными пакетами винчестерский диск (или пакет дисков) нельзя снять с дисковода и положить на полку, поэтому его нельзя использовать для хранения резервных копий данных. Для получения резервной копии с винчестерского диска используется множество дискет или магнитных лент.

Накопители на магнитных лентах разделяются на катушечные (НМЛ) и кассетные (КНМЛ). Это устройства с последовательным (более длительным) доступом к информации (сравните время доступа к нужной записи песни на магнитофоне и грампластинке – аналоге НМД).

В микроЭВМ применяют преимущественно КНМЛ, использующие компакт-кассеты (рис. 7.21). Это могут быть обычные бытовые кассетные магнитофоны или специально изготовленные КНМЛ, в которых учитывается специфика записываемой цифровой информации. При записи на бытовой кассетный магнитофон, предназначенный для работы с сигналами звукового диапазона, часто используют частотную модуляцию (см. рис. 7.3, *в*), при которой, например, 1200 Гц соответствуют 0, а 2400 Гц – 1. Такое кодирование, являясь самосинхронизирующимся, позволяет компенсировать изменение скорости движения ленты. При записи в последовательном коде (см. рис. 7.1) с плотностью около 6 бит/мм на одной дорожке ленты длиной 90 м можно разместить 50 Кбайт информации (скорость обмена данными 300 бит/с). Использование специальных цифровых КНМЛ с двухдорожечной (или четырехдорожечной) записью и плотностью до 63 бит/мм увеличивает емкость такого ЗУ до нескольких мегабайтов, а скорость обмена – до 0,5 Кбайт/с.

Разработаны цифровые КНМЛ с другими типами кассет. Это и малогабаритные кассеты с бесконечной лентой, что упрощает механизм накопителя,

и кассеты с шириной ленты 6,35 мм и длиной от 45 до 180 м, где на двух или четырех дорожках размещаются до нескольких десятков мегабайтов информации. Такие накопители используют для копирования винчестерских дисков.

Аналогичную функцию возлагают и на катушечные НМЛ с лентой шириной 12,7 мм (рис. 7.22). В этих НМЛ информация записывается на девяти дорожках, т. е. поперек ленты записывается один байт, и одна дорожка (расположенная так же, как и синхродорожка перфоленты на рис. 7.5) предназначена для контроля при считывании информации и для обеспечения синхронизации. Обычно в таких НМЛ используется контроль по нечетности (если число единиц байта четно, то на этой дорожке записывается единица), т. е. при чтении с любой строки считается хотя бы один единичный сигнал, информирующий о наличии этой строки (самосинхронизация).

Байты, записанные на ленте, группируются в блоки данных (или зоны) переменной длины. В начале и конце блоков содержатся идентифицирующая и управляющая информация, а также контрольные коды (байт продольного контроля, биты которого дополняют сумму единиц по соответствующим дорожкам до четного числа, и байт циклического контроля, формируемый в процессе записи устройством управления НМЛ).

Блоки разделяются межзонными промежутками для исключения потери информации при стартстопной работе с НМЛ. На типичной катушке диаметром 267 мм с длиной ленты 730 м при плотности записи 32 бит/мм можно записать по 15 Мбайт информации, а при увеличении плотности записи до 63 или 246 бит/мм и обмене непрерывным потоком (без стартстопа между блоками) — до 180 Мбайт. Скорость обмена информации может быть доведена в таких устройствах до 0,5 Мбайт/с.

Память на цилиндрических магнитных доменах (ЦМД) занимает промежуточное положение между ОЗУ и НМЛ.

Некоторые материалы обладают тем свойством, что их тонкие пленки на подложке содержат случайно распределенные магнитные домены (буквально — пузырьки), одноосные поля которых перпендикулярны плоскости пленки. При подаче внешнего магнитного поля с направлением, противоположным полю доменов, домен может сжаться в маленький цилиндр (диаметром от 1 до 10 мкм)

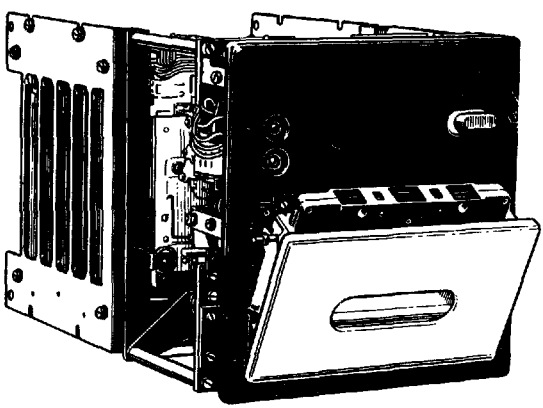


Рис. 7.21. Кассетный накопитель на магнитной ленте (КНМЛ) «Роботрон К-5200»

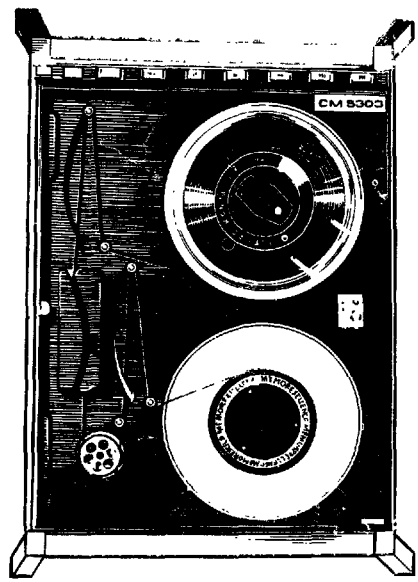


Рис. 7.22. Накопитель на магнитной ленте (НМЛ) СМ-5303

или исчезнуть совсем. При подаче магнитного поля, направленного в плоскости пленки, магнитные домены можно передвигать в пленке.

Если к пленке приложен пермаллоевый шаблон, домены можно заставить двигаться по направлениям, определяемым шаблоном. Если, наконец, между пленкой и шаблоном поместить проводящую сетку, токи в сетке можно использовать для локального управления магнитным полем так, что отдельные домены можно генерировать, аннигилировать, увеличивать в размерах и расщеплять (дублировать) или заставлять перемещаться. Таким образом, появляется возможность построить память в виде длинного регистра сдвига (или нескольких регистров), в котором наличие домена означает 1, а отсутствие — 0. Как и все магнитные устройства, такая память оказывается энергонезависимой.

ЦМД-память выпускают в виде БИС емкостью от 64 Кбит до 1 Мбит. Время доступа к информации, расположенной в памяти, построенной из таких БИС, на порядок меньше, чем у винчестерских дисков. Однако пока эта память еще достаточно громоздка и дорога, хотя ее высокая надежность (отсутствие подвижных механических частей) заставляет специалистов искать пути снижения стоимости и увеличения емкости БИС ЦМД-памяти.

Из немагнитных внешних запоминающих устройств наиболее интересными являются оптические диски. Они обладают большой емкостью памяти в расчете на один диск (примерно 200 Мбайт для формата 133 мм) и низкой удельной стоимостью одного бита хранимой информации. Эти диски надежны в эксплуатации, легко транспортируются и не требуют столь деликатного обхождения, как магнитные носители. Двоичные комбинации единиц и нулей записываются на оптический диск лазером, который служит и для их считывания, но при меньшем уровне интенсивности излучения.

Однако пока допускается лишь однократная запись информации на диск, поэтому они используются для ведения архивной памяти, например, в системах медицинского обслуживания, образования. Ожидается, что со временем на оптических дисках удастся реализовать весь набор функций считывания/записи и они будут способны конкурировать с винчестерскими дисками.

И в заключение рассмотрим эмуляторы внешних запоминающих устройств. В ряде случаев (например, когда не хватает основной памяти машины и эта память уже не может расширяться при существующей разрядности счетчика команд микроЭВМ) прибегают к использованию электронного диска — полупроводниковой памяти (ОЗУ, ПЗУ, ППЗУ или СППЗУ) с устройством управления, позволяющим организовать между ней и микроЭВМ последовательный обмен данными через контроллер НМД. МикроЭВМ обманывают: она «думает», что обменивается данными с очень быстродействующим НМД (время доступа на два порядка выше, чем у винчестерских дисков).

Такая имитация какой-либо системы получила название эмуляции (emulate — соревнование).

Подобным образом (меняя интерпретатор микропрограммного устройства управления) из существующей микроЭВМ создают такую, которая имеет систему команд еще не реализованной ЭВМ (для подготовки программного обеспечения новой ЭВМ) или очень старой ЭВМ (чтобы продлить «жизнь» ее программ).

Недостаток эмуляторов накопителей на магнитных дисках — их энергозависимость: при выключении питания информация стирается с такого диска.

Глава 8

ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА ИНФОРМАЦИИ

8.1. Интерфейсы микроЭВМ

В гл. 3 определена типичная структура системы ввода-вывода микроЭВМ и кратко описаны функции ее составных частей.

Конкретная реализация системы ввода-вывода — номенклатура шин в интерфейсах системы ввода-вывода, типы контроллеров ВУ, способы передачи информации по шинам интерфейса (параллельная или последовательная передача, синхронная или асинхронная) — определяется в первую очередь назначением микроЭВМ в целом. Каким же образом?

Назначение (область применения) микроЭВМ, т. е. класс реализуемых алгоритмов, обусловливает, во-первых, выбор типа центрального процессора и, во-вторых, количество и перечень требуемых ВУ и каналов связи. В соответствии с этими двумя факторами в системах ввода-вывода большинства современных микроЭВМ можно выделить два уровня сопряжения ВУ с процессором и памятью. На первом уровне контроллеры ВУ сопрягаются с процессором и памятью через системный интерфейс микроЭВМ, который обеспечивает комплексирование отдельных устройств микроЭВМ в единую систему. На втором уровне сопряжения контроллеры посредством шин связи с ВУ соединяются с соответствующими внешними устройствами микроЭВМ.

На первом уровне сопряжения набор шин интерфейса ввода-вывода и алгоритм его функционирования полностью определяются типом БИС процессора — его системным интерфейсом. Несмотря на широкое разнообразие системных интерфейсов микроЭВМ, в общем случае можно выделить два основных способа использования системного интерфейса для организации обмена информацией с ВУ: 1) с применением специальных команд ввода-вывода; 2) по аналогии с обращениями к памяти.

Рассмотрим, как используются для обмена информацией с ВУ шины адреса и данных системных интерфейсов микроЭВМ и какие необходимы для этого управляющие сигналы.

При использовании для обмена с ВУ команд ввода-вывода адрес (номер) ВУ передается по шине адреса. Однако по этой же шине передаются и адреса ячеек памяти. Информация на шине адреса имеет смысл адреса (номера) ВУ только при наличии специальных управляющих сигналов. Такими сигналами могут быть, например, «Ввод из ВУ» и «Вывод в ВУ» (рис. 8.1, *а*), инициируемые соответствующими командами ввода-вывода микроЭВМ.

Для синхронизации работы процессора микроЭВМ и контроллеров ВУ, а точнее, для указания моментов времени, определяющих готовность данных в ВУ для передачи либо подтверждающих их прием, может служить управляющий осведомительный сигнал «Готовность ВУ».

Такого простого набора управляющих сигналов в общем случае достаточно для организации программно-управляемого обмена данными с ВУ на первом уровне (процессор — контроллер ВУ). Порядок использования описанных выше управляющих сигналов при выполнении операций ВВОД и ВЫВОД проиллюстрирован рис. 8.2.

Операция ВЫВОД, инициируемая соответствующей командой микропроцессора, выполняется следующим образом. Микропроцессор выставляет на линиях адресной шины адрес (номер) ВУ, на линиях шины данных —

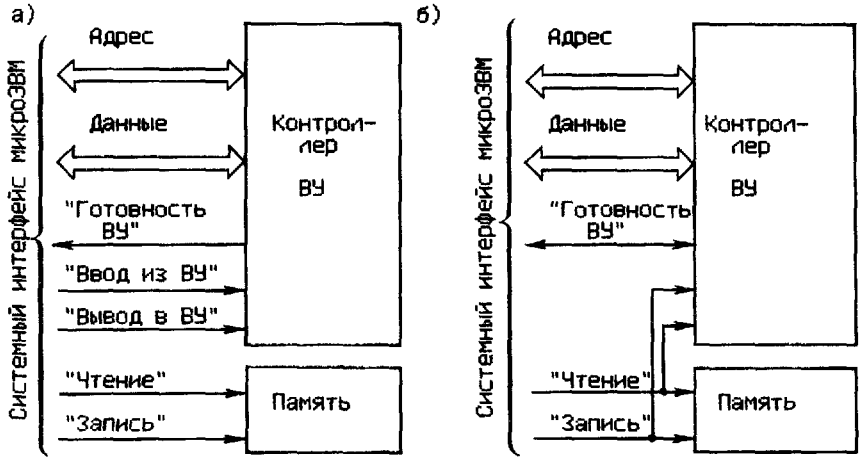


Рис. 8.1. Простейшее сопряжение контроллера ВУ с системным интерфейсом

значения разрядов выводимого слова данных и единичным сигналом по линии «Вывод в ВУ» указывает тип операции. Адресуемый контроллер ВУ принимает данные, пересылает их в ВУ и единичным сигналом по линии «Готовность ВУ» сообщает процессору, что данные приняты ВУ и можно снять информацию с шин адреса и данных, а также сигнал «Вывод в ВУ».

Выполнение операции ВВОД (рис. 8.2, *б*) начинается с того, что микропроцессор выставляет на линиях адресной шины адрес (номер) ВУ и единичным сигналом на линии «Ввод из ВУ» указывает тип выполняемой операции. По сигналу «Ввод из ВУ» контроллер адресуемого ВУ считывает слово данных из ВУ, выставляет на линиях шины данных значения разрядов считанного слова и единичным сигналом по линии «Готовность ВУ» сообщает об этом процессору. Приняв данные из контроллера ВУ, процессор снимает сигналы с шины адреса и линии «Ввод из ВУ»

При реализации в микроЭВМ обмена с ВУ по аналогии с обращениями к памяти отпадает необходимость в специальных сигналах, указывающих,

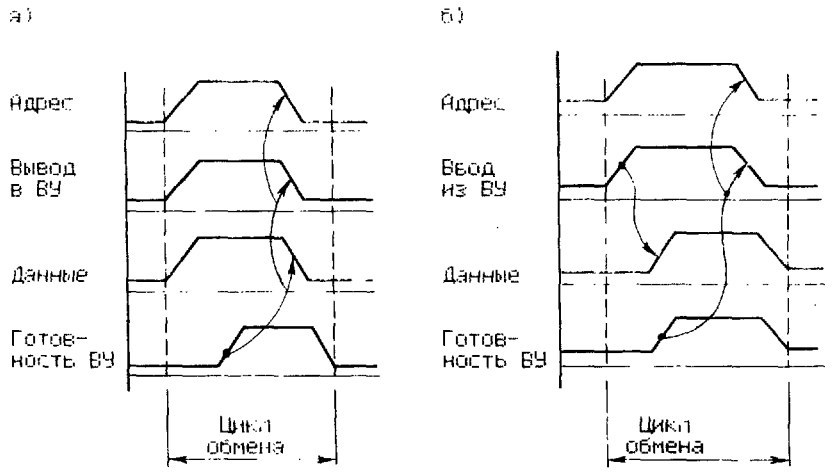


Рис. 8.2. Временные диаграммы операций для простейшего набора управляющих сигналов:

а - ВЫВОД; *б* - ВВОД

что на шине адреса находится адрес ВУ. Для адресов ВУ отведена часть адресного пространства микроЭВМ, и в контроллерах ВУ удается легко организовать селекцию адресов ВУ — выделение собственного адреса ВУ из всего множества адресов, передаваемых по линиям адресной шины. Однако остается необходимость передавать в ВУ приказ на ввод или вывод информации. Для этих целей используются линии управляющей шины «Чтение» и «Запись», обеспечивающие обмен информацией микропроцессора с модулями памяти (см. рис. 8.1, *б*).

Временные диаграммы операций ВВОД и ВЫВОД при организации обмена с ВУ по аналогии с памятью полностью соответствуют рассмотренным ранее (рис. 8.2).

Приведенный здесь минимальный набор управляющих (осведомительных) сигналов позволяет организовать обмен с ВУ не только в асинхронном режиме, но и в режиме прерывания программы. Правда, при этом существенно усложняются алгоритмы использования управляющих сигналов и, как следствие, аппаратура для их обработки. Более рациональным оказывается увеличение числа управляющих сигналов с тем, чтобы каждый режим обмена идентифицировался отдельным сигналом или набором сигналов. Для этого в системных интерфейсах вводятся линии для передачи сигналов запроса на прерывание и предоставления прерывания, запроса на предоставление прямого доступа к памяти и его предоставления и т. п.

Но в микроЭВМ необходимо управлять еще и режимами работы контроллеров разнообразных ВУ, многие из которых являются достаточно сложными устройствами. При этом каждый контроллер воспринимает определенный, присущий только данному ВУ набор команд управления (приказов ВУ). Организовать в этом случае передачу каждого приказа ВУ по отдельной линии системного интерфейса не представляется возможным по двум причинам. Во-первых, при разработке микропроцессора и системного интерфейса достаточно трудно предусмотреть все возможные применения микроЭВМ на его основе, а следовательно, и используемые в микроЭВМ ВУ. И во-вторых, для каждого дополнительного управляющего сигнала потребуется отдельный вывод в БИС микропроцессора. Таким образом, возникают чисто конструктивные ограничения на количество используемых в системном интерфейсе управляющих сигналов, связанных с числом выводов в БИС микропроцессора.

Решение указанной проблемы осуществляется путем мультиплексирования шины данных (см. параграф 5.2), т. е. использования ее для обмена с контроллерами ВУ как данными (в одни моменты времени), так и частью управляющей информации (в другие моменты времени). Однако пересылаемая информация должна размещаться в различных регистрах контроллера ВУ: данные — в регистре данных, а управляющая информация — в одном или нескольких регистрах состояния и управления (количество этих регистров возрастает с увеличением сложности ВУ и уменьшением разрядности передаваемых слов, т. е. ширины шины данных). Это ставит новую задачу — выбор одного или нескольких регистров контроллера ВУ.

Наиболее просто эта задача решается выделением каждому регистру контроллера собственного адреса в системе адресов микроЭВМ. В этом случае организация обмена информацией микропроцессора с регистрами контроллеров не потребует дополнительных линий системного интерфейса, так как для адресации регистров используется шина адреса. Таким образом, в микроЭВМ каждому ВУ выделяется столько адресов, сколько регистров в его контроллере.

Рассмотрим конкретные примеры реализации системных интерфейсов микроЭВМ и их использования для обмена с ВУ. Типичными и наиболее

распространенными системными интерфейсами микроЭВМ, в которых реализуются различные способы обмена с ВУ (с использованием команд ввода-вывода и по аналогии с обращением к памяти), являются системные интерфейсы МПК БИС КР580 (Microbus) и микроЭВМ «Электроника-60» (Q-bus).

Номенклатура линий системного интерфейса Microbus определяется выводами БИС МП КР580ИК80А (рис. 8.3). Всего в интерфейсе исполь-

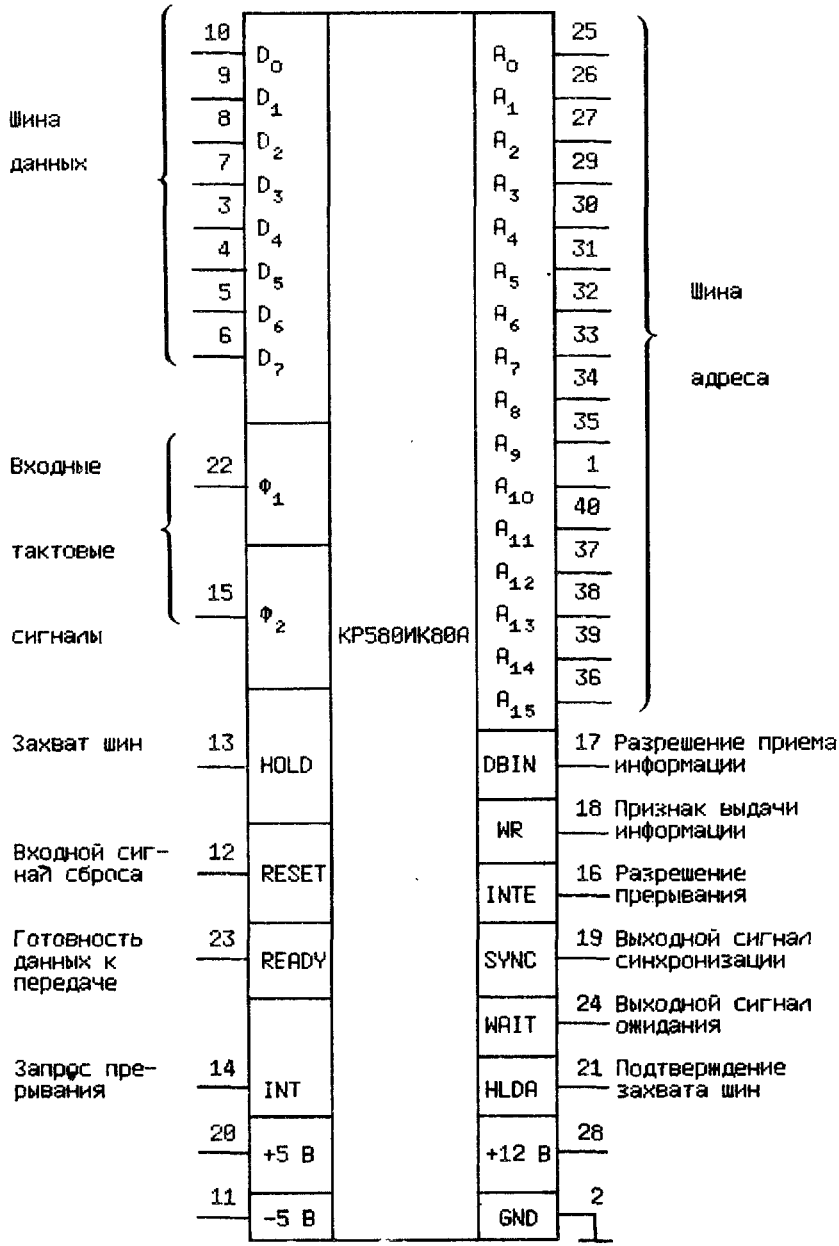


Рис. 8.3. Обозначение и нумерация выводов БИС КР580ИК80А

зуются 36 линий, которые разделены на три шины (см. также рис. 5.2, *б*):

16-разрядная шина адреса $A_0, \ldots, A_{15}$;

8-разрядная шина данных $D_0, \ldots, D_7$;

12-разрядная шина управления.

Шина адреса предназначена, во-первых, для пересылки 16-разрядного адреса ячейки памяти, что обеспечивает прямую адресацию памяти емкостью 64 Кбайт, и, во-вторых, для передачи по восьми младшим линиям ША0, ... , ША7 адреса ВУ, что обеспечивает обращение к 256 регистрам контроллеров ВУ при выполнении команды ВЫВОД (OUT) и к 256 регистрам контроллеров ВУ при выполнении команды ВВОД (IN).

Шина данных интерфейса Microbus используется в трех различных режимах:

1) для обмена байтом информации между процессором и памятью, т. е. при чтении из памяти команды или данных и при записи в память данных;

2) для обмена байтом данных между процессором и контроллером ВУ при выполнении команд ВВОД (IN) и ВЫВОД (OUT);

3) для передачи 8-разрядного слова состояния MII (управляющего слова) во внешние схемы управления работой микроЭВМ.

Необходимость передачи части управляющих сигналов по шине данных возникла из-за малого числа выводов БИС МП, не позволяющего выделить для шины управления более 12 выводов.

Шина управления предназначена для передачи шести входных и шести выходных сигналов. Вместе со словом состояния МП они обеспечивают как управление работой процессора в микроЭВМ, так и порядок использования шин адреса и данных при взаимодействии процессора с памятью и ВУ. Использование передаваемого по шине данных управляющего слова позволяет организовать эффективное управление работой системы ввода-вывода микроЭВМ при ограниченном количестве выводов БИС микропроцессора наборе управляющих сигналов. Наименование разрядов слова состояния MII, их значения для машинных циклов, использующихся при реализации операций ввода-вывода, и описания управляющих сигналов приведены соответственно в табл. 8.1 и 8.2.

Управляющие сигналы интерфейса Microbus позволяют организовать в микроЭВМ на его основе два режима обмена информацией с контроллерами ВУ:

программно-управляемый (асинхронный и по прерыванию программы);

в режиме прямого доступа в память.

Таблица 8.1

Слово состояния микропроцессора КР580ИК80А при вводе-выводе информации

Номер разряда	Обозначение	Наименование сигнала	Значение разрядов D_0 D_7			
			Название машинного цикла			
			Ввод по каналу ввода	Вывод по каналу вывода	Прерывание	Прерывание во время останова
D_0	INTA	Подтверждение прерывания	0	0	1	1
D_1	$\overline{WO}$	Запись или вывод	1	0	1	1
D_2	STACK	Операция со стеком	0	0	0	0
D_3	HLTA	Подтверждение останова	0	0	0	1
D_4	OUT	Вывод	0	1	0	0
D_5	M1	Первый цикл команды	0	0	1	1
D_6	INP	Ввод	1	0	0	0
D_7	MEMR	Чтение из памяти	0	0	0	0

Подробнее организация различных типов обмена будет рассмотрена в следующих параграфах главы. А здесь следует отметить лишь одну особенность в организации обмена с ВУ в микроЭВМ на МПК БИС КР580: алгоритм функционирования интерфейса Microbus при обмене МП с ВУ допускает обработку в МП сигналов запроса прерывания и захвата шин от ВУ во время останова МП. За счет этого удается организовать эффективное взаимодействие программ микроЭВМ и медленнодействующего периферийного оборудования микроЭВМ в системах реального времени.

Системный интерфейс микроЭВМ «Электроника-60» (Q-bus) состоит из 34 линий, которые разделены на две шины (см. рис. 5.2, *в*):

16-разрядная мультиплексируемая шина (см. параграф 5.2) «адрес/данные» ДА00, ... , ДА15;

18-разрядная шина управления.

Шина «адрес/данные» состоит из 16 двунаправленных линий, которые предназначены для поочередной передачи адресов (ячеек памяти или ВУ) и данных. В микроЭВМ на основе интерфейса Q-bus обращения к ВУ производятся аналогично обращениям к памяти. Для этого часть адресного пространства микроЭВМ, начиная с адреса $(160000)_8$ до адреса $(177776)_8$ (всего 4096 адресов), выделена для адресов регистров контроллеров ВУ. Поэтому 16-разрядная шина обеспечивает прямую адресацию памяти емкостью 56 Кбайт и 4096 (4К) регистров контроллеров ВУ. Обмен данными по шине «адрес/данные» может производиться как 16-разрядными словами, так и байтами. Для передачи данных байтами используются восемь линий шины ДА00, ... , ДА07.

Таблица 8.2

Управляющие сигналы интерфейса Microbus

Обозначение сигнала	Наименование сигнала	Источник сигнала	Назначение сигнала
DBIN	Разрешение приема информации с шины данных	МП	Шина данных $D_0,...,D_7$ МП находится в режиме приема информации
WR	Признак выдачи информации на шину данных	МП	На шину данных $D_0,...,D_7$ МП выдана информация для передачи в память или в ВУ
READY	Готовность данных к передаче	Память, ВУ	Информация в памяти или в ВУ готова для передачи в МП. При отсутствии сигнала READY МП переходит в состояние ожидания
WAIT	Ожидание	МП	Микропроцессор находится в состоянии ожидания, выходит из него при поступлении сигнала READY
INT	Запрос прерывания	ВУ	Запрос ВУ на обслуживание прерывание текущей программы после выполнения очередной команды, запрос ВУ на обслуживание в состоянии ожидания МП во время останова
INTE	Разрешение прерывания	МП	Разрешение ВУ на обмен информацией в режиме прерывания программы
HOLD	Захват шин	ВУ	Запрос ВУ на прямой доступ к памяти
HLDA	Подтверждение захвата шин	МП	Разрешение ВУ на обмен с памятью в режиме прямого доступа. Выходы шин адреса и данных МП переводятся в состояние с высоким выходным сопротивлением
SYNC	Синхронизация	МП	Начало машинного цикла. Подтверждает выдачу микропроцессором на шину данных слова состояния управляющего слова
RESET	Сброс	Внешние схемы управления	Устанавливает в нуль счетчик команд, триггеры разрешения прерывания и захвата шин

Сигналы, передаваемые по линиям шины управления, обеспечивают управление:

передачей информации по шине «адрес/данные»;

режимами обмена с ВУ с учетом их приоритетов;

работой микроЭВМ в целом.

Управление передачей информации (адресов или данных) сводится к реализации стандартных циклов обмена (ВВОД или ВЫВОД) между двумя устройствами, подключенными к системному интерфейсу. На одно из этих устройств возлагаются функции по управлению обменом (по реализации цикла обмена), и оно называется активным. Активным устройством обычно является процессор микроЭВМ, но это может быть и контроллер прямого доступа к памяти. Второе устройство, участвующее в обмене, называется пассивным. Так, например, память микроЭВМ всегда является пассивным устройством при обмене информацией. Для реализации стандартных циклов обмена активным и пассивным устройствами используется следующий набор управляющих сигналов: СИА, СИП, ВВОД, ВЫВОД, БАЙТ, ВУ.

Сигнал синхронизации активного устройства (СИА) вырабатывается активным устройством. Передний фронт этого сигнала указывает, что на линиях ДА00, ... , ДА15 активным устройством выставлены адрес ВУ или ячейки памяти. Сигнал сохраняет единичное значение в течение всего цикла обмена.

Сигнал синхронизации пассивного устройства (СИП) извещает активное устройство о том, что данные приняты с линий ДА00, ... , ДА15 при выполнении операций ВЫВОД или данные установлены пассивным устройством на линиях ДА00, ... , ДА15 при выполнении операции ВВОД. Этот сигнал вырабатывается пассивным устройством в ответ на сигналы активного устройства ВВОД и ВЫВОД.

Сигнал ВВОД вырабатывается активным устройством во время действия сигнала СИА и означает, что выполняется ввод данных в активное устройство и оно готово принять данные.

Сигнал ВЫВОД вырабатывается активным устройством и означает, что на линиях ДА00, ... , ДА15 активным устройством выставлены данные. Пассивное устройство принимает данные и в подтверждение этого должно выработать сигнал СИП.

Сигнал БАЙТ используется только в цикле ВЫВОД и указывает, что активным устройством выводится один байт информации по линиям ДА00, ... , ДА07.

Сигнал ВУ вырабатывается активным устройством в том случае, если на линиях ДА00, ... , ДА15 установлен адрес, относящийся к старшим 4К адресов (с 160000_8 по 177776_8). Введение этого сигнала позволяет сократить затраты оборудования, используемого в контроллерах ВУ для селекции адресов, передаваемых по системному интерфейсу.

Рассмотренные управляющие сигналы обеспечивают реализацию в микроЭВМ на основе системного интерфейса Q-bus асинхронного программно-управляемого обмена данными с ВУ.

Описания управляющих сигналов, обеспечивающих обмен по прерыванию программы и обмен в режиме прямого доступа в память, приведены в табл. 8.3.

Сигналы, управляющие работой микроЭВМ в целом, обеспечивают начальную установку всех устройств микроЭВМ (СБРОС), регенерацию динамической памяти микроЭВМ (РГН), перевод процессора в останов и режим связи с пультовым терминалом (ОСТ), извещают процессор о состоянии источников питания (ПОСТН, ПИТН).

Таблица 8.3

Сигналы интерфейса Q-bus, управляющие режимами обмена с ВУ с учетом их приоритетов

Обозначение сигнала	Наименование сигнала	Источник сигнала	Назначение сигнала
ТПР	Требование прерывания	ВУ	Информирует процессор, что ВУ готово к обмену данными. Вызывает прерывание программы после выполнения очередной команды
ППР	Предоставление прерывания	Процессор	Извещает ВУ, что прерывание разрешено. Линия ППР проходит последовательно через все ВУ, обеспечивая приоритетное обслуживание запросов на прерывание
ТПД	Требование прямого доступа к памяти	ВУ	Информирует процессор, что ВУ требует обмена в режиме прямого доступа к памяти
ППД	Предоставление прямого доступа к памяти	Процессор	Разрешение ВУ на обмен с памятью в режиме прямого доступа. Процессор приостанавливает выполнение команды после завершения текущего цикла обмена. Линия ППД проходит через ВУ аналогично линии ППР
ПВ	Подтверждение выбора	ВУ	Вырабатывается в ответ на сигнал ППД и подтверждает, что ВУ ведет обмен с памятью в режиме прямого доступа
ПРТ	Требование прерывания по внешнему событию	Аппаратный таймер, ВУ	Обычно используется для передачи в процессор сигналов аппаратного таймера. Имеет приоритет выше, чем сигнал ТПР

В следующих параграфах главы рассмотрим, как с использованием управляющих сигналов системных интерфейсов в микроЭВМ организуется обмен данными с различными ВУ.

8.2. Контроллеры внешних устройств

Подключение любого внешнего устройства к микроЭВМ осуществляется через контроллер ВУ. Способы структурной и функциональной организации контроллеров ВУ определяются двумя основными факторами:

форматами данных и режимами работы конкретных ВУ;

типом системного интерфейса микроЭВМ.

Влияние первого из этих факторов на организацию контроллеров ВУ достаточно очевидно. В самом деле, нерационально было бы создавать единый универсальный контроллер, обеспечивающий, например, подключение к микроЭВМ простых устройств типа цифровых индикаторов и сложных устройств типа накопителей на магнитных дисках. Именно поэтому на практике применяют самые разнообразные контроллеры ВУ: от простейших для подключения к микроЭВМ датчиков одиночных сигналов до очень сложных, сравнимых по сложности с процессорами микроЭВМ, например, для сопряжения микроЭВМ с аппаратурой для автоматизации научных исследований в стандарте КАМАК.

Второй фактор — тип интерфейса — определяет способ организации электронных схем контроллеров ВУ, обеспечивающих связь с шинами интерфейса, в первую очередь — схем распознавания адресов ВУ. Наиболее просто дешифрация адреса ВУ реализуется в контроллерах ВУ для микроЭВМ с отдельным интерфейсом ввода-вывода (см. рис. 5.2, *а*). Более сложны схемы распознавания адресов и разделения адресов и данных, передаваемых по единым шинам, в контроллерах ВУ, применяемых в микроЭВМ со структурами, изображенными на рис. 5.2, *г* и *д* соответственно.

Однако такое положение дел, когда для каждого типа ВУ требуется уникальный контроллер, конечно, не устраивало ни разработчиков средств вычислительной техники (микроЭВМ и ВУ), ни ее пользователей. Наиболее перспективным оказался путь стандартизации набора информационных и управляющих сигналов, которыми обмениваются контроллер и ВУ. Для разных типов микроЭВМ были разработаны контроллеры, обеспечивающие:

связь с ВУ по стандартному параллельному (ИРПР) каналу передачи данных;

связь с ВУ по стандартному последовательному (ИРПС) каналу передачи данных;

преобразование информации из аналоговой формы в цифровую с заданной точностью;

преобразование информации из дискретной формы представления в аналоговую в заданных диапазонах изменения аналоговых величин.

Рассмотрим типичные структуры контроллеров ВУ, применяемых в микроЭВМ с различными системными интерфейсами. На рис. 8.4, *а* приведена блок-схема типичного контроллера ВУ, обеспечивающего программно-управляемый обмен информацией с ВУ (операции ВВОД и ВЫВОД, см. рис. 8.2) в микроЭВМ, имеющей системный интерфейс с изолированными шинами адреса и данных.

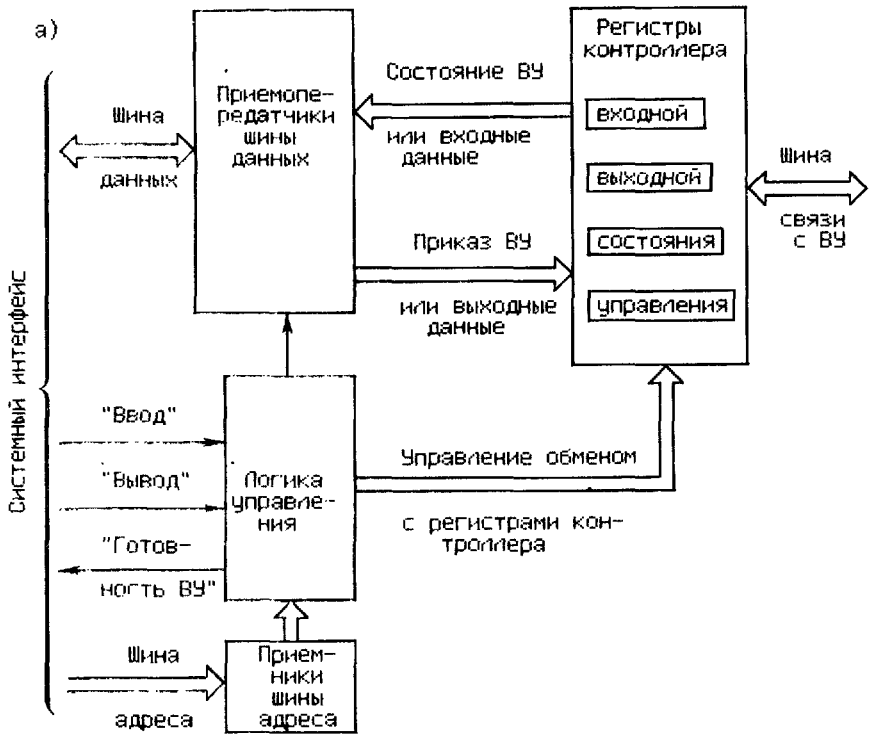
Основу контроллера ВУ составляют несколько регистров, которые служат для временного хранения передаваемой информации. Каждый регистр имеет свой адрес, и зачастую такие регистры называют портами ввода-вывода. Регистры входных и выходных данных работают соответственно только в режиме чтения и только в режиме записи. Регистр состояния работает только в режиме чтения и содержит информацию о текущем состоянии ВУ (включено/выключено, готово/не готово к обмену данными и т. п.). Регистр управления работает только в режиме записи и служит для приема из микроЭВМ приказов для ВУ. В контроллерах, используемых для подключения достаточно простых ВУ (клавиатура, перфоратор и т. п.), удается совместить в один регистры состояния и управления, что позволяет сократить количество используемых в контроллере портов ввода-вывода, а следовательно, и адресов, выделенных для данного ВУ.

Логика управления контроллера ВУ выполняет селекцию адресов регистров контроллера, прием, обработку и формирование управляющих сигналов системного интерфейса, обеспечивая тем самым обмен информацией между регистрами контроллера и шиной данных системного интерфейса микроЭВМ.

Приемопередатчики шин адреса и данных служат для физического подключения электронных схем контроллера к соответствующим шинам системного интерфейса.

На рис. 8.4, *б* приведена блок-схема типичного контроллера ВУ для системного интерфейса с мультиплексируемой шиной «адрес/данные». Сравнение двух контроллеров на рис. 8.4, *а* и *б* показывает, что между ними нет принципиальных различий в порядке использования регистров. Различия в структурах контроллеров, вызванные различной организацией системных интерфейсов, проявляются только в построении логики управления (поразному организованы прием и селекция адресов) и способе подключения к шинам системного интерфейса.

В настоящее время в практику широко внедряются программируемые контроллеры, режимы работы которых устанавливаются специальными командами микроЭВМ или определяются программами обмена с ВУ, хранимыми, как правило, в сменных БИС ПЗУ или ППЗУ. Программируемые контроллеры необходимо настраивать на конкретный режим обмена дан-



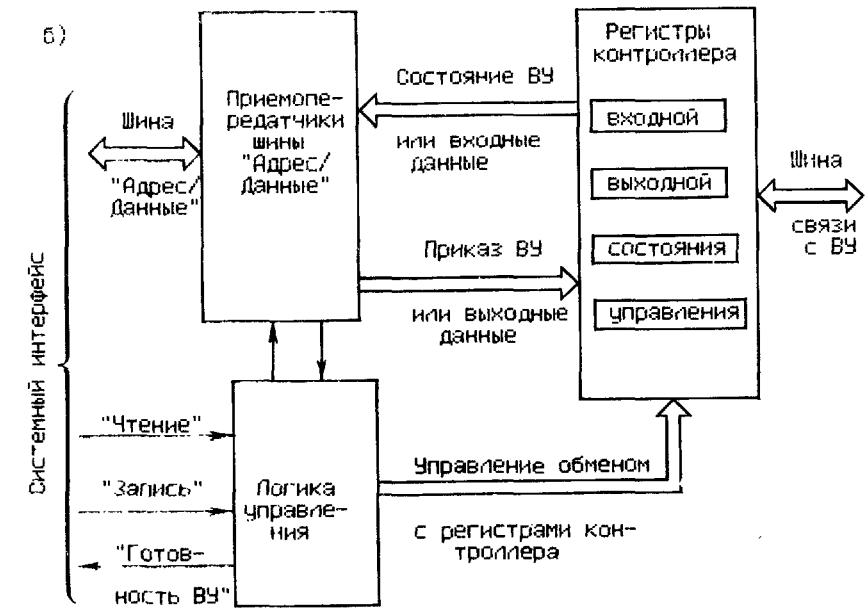


Рис 8.4. Блок-схема типичного контроллера ВУ для системного интерфейса: *а* — с раздельными шинами адреса и данных; *б* — с мультиплексированной шиной «Адрес/Данные»

ными, присущий ВУ: синхронный или асинхронный, с использованием сигналов прерывания или без их использования, на заданную скорость обмена и т. д. Настройка таких контроллеров на требуемый режим обмена производится программным путем с помощью специальных команд (управляющих слов), передаваемых из процессора в контроллер ВУ перед началом обмена. Управляющее слово записывается в специальный регистр и инициирует заданный режим обмена с ВУ.

Развитие интегральной микросхемотехники позволило создавать программируемые контроллеры в виде одной БИС. В качестве примера можно привести БИС КР580ВВ51 — программируемый последовательный интерфейс и БИС КР580ВВ55 — программируемый параллельный интерфейс.

Для подключения к микроЭВМ ВУ со сложными алгоритмами управления (например, накопителей на магнитных дисках) используются программируемые контроллеры с хранимыми в ПЗУ или ППЗУ программами

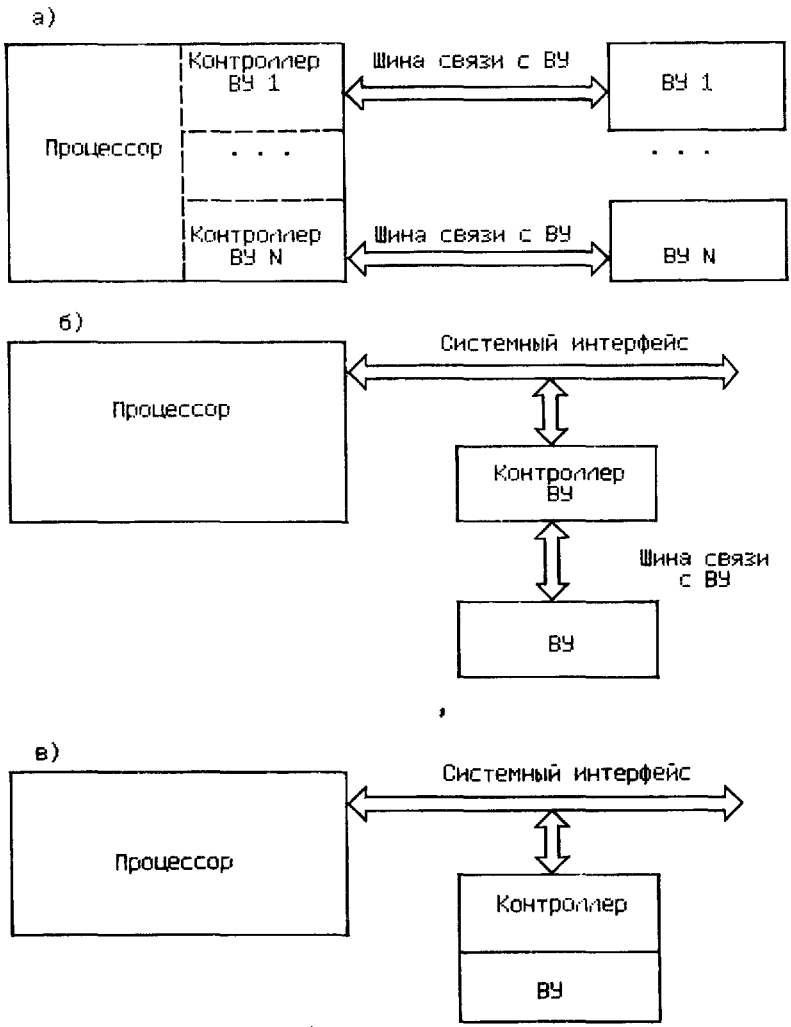


Рис. 8.5. Различные варианты реализации контроллеров ВУ:

а — на плате процессора; *б* — в виде отдельного устройства; *в* — непосредственно в ВУ

управления и обмена с ВУ. Такого рода контроллеры являются, по сути дела, специализированными микроЭВМ, а их настройка на взаимодействие с конкретным ВУ осуществляется сменой программы, т. е. заменой БИС ПЗУ или ППЗУ.

В современных микроЭВМ зачастую достаточно трудно провести границу между интерфейсом ввода-вывода и контроллерами ВУ. Это проявляется даже в том, что области использования терминов «интерфейс ввода-вывода» и «контроллер» не являются полностью установившимися. Например, в составе микроЭВМ «Электроника-60» есть контроллер (плата И2), обеспечивающий связь с ВУ по стандартному параллельному каналу передачи данных; называется этот контроллер «параллельный интерфейс». Стандартизация интерфейсов ввода-вывода и использование БИС для реализации логических схем интерфейсов и контроллеров ВУ позволяет конструктивно реализовать контроллер (интерфейс ввода-вывода) на плате процессора либо непосредственно в ВУ (рис. 8.5). Такая реализация интерфейсов и контроллеров создает у пользователя иллюзию их отсутствия и некоторую несогласованность в структурных и функциональных схемах микроЭВМ. Тем не менее в любой микроЭВМ всегда можно достаточно четко выделить компоненты системы ввода-вывода, что крайне необходимо при изучении как микроЭВМ, так и способов их использования в различных системах.

В следующих параграфах главы при описании конкретных способов обмена данными с ВУ будут рассмотрены и контроллеры ВУ, обеспечивающие их реализацию.

8.3. Параллельная передача данных

Параллельная передача данных между контроллером и ВУ является по своей организации наиболее простым способом обмена. Для организации параллельной передачи данных помимо шины данных, количество линий в которой равно числу одновременно передаваемых битов данных, используется минимальное количество управляющих сигналов.

В простом контроллере ВУ, обеспечивающем побайтную передачу данных во внешнее устройство (рис. 8.6), в шине связи с ВУ используются всего два управляющих сигнала — «Выходные данные готовы» и «Данные приняты».

Для формирования управляющего сигнала «Выходные данные готовы» и приема из ВУ управляющего сигнала «Данные приняты» в контроллере используется одноразрядный адресуемый регистр состояния и управления *A2*. Одновременно с записью очередного байта данных из шины данных системного интерфейса в адресуемый регистр данных контроллера (порт вывода *A1*) в регистр состояния и управления записывается логическая единица. Тем самым формируется управляющий сигнал «Выходные данные готовы» в шине связи с ВУ.

ВУ, приняв байт данных, управляющим сигналом «Данные приняты» обнуляет регистр состояния контроллера. При этом формируются:

управляющий сигнал системного интерфейса «Готовность ВУ»;

признак готовности ВУ к обмену, передаваемый в процессор по одной из линий шины данных системного интерфейса посредством стандартной операции ввода при реализации программы асинхронного обмена.

Логика управления контроллера обеспечивает селекцию адресов регистров контроллера, прием управляющих сигналов системного интерфейса и формирование на их основе внутренних управляющих сигналов контроллера, формирование управляющего сигнала системного интерфейса «Готовность ВУ». Для сопряжения регистров контроллера с шинами адреса и

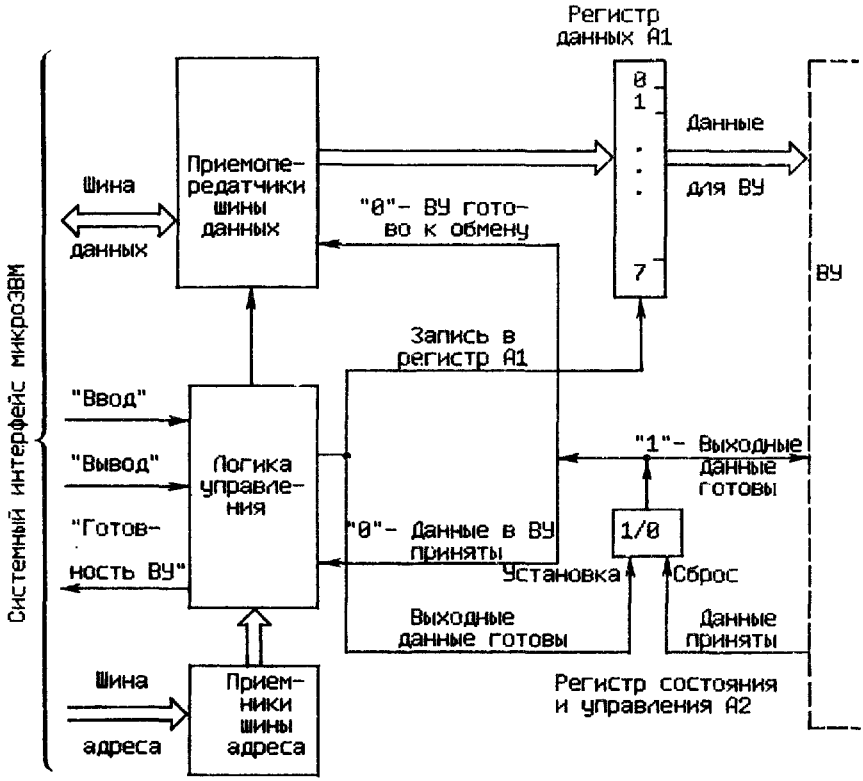


Рис. 8.6. Простой параллельный интерфейс (контроллер) вывода

данных системного интерфейса в контроллере используются приемники шины адреса и приемопередатчики шины данных.

Рассмотрим на примере, каким образом контроллер ВУ обеспечивает параллельную передачу данных в ВУ под управлением программы асинхронного обмена. Алгоритм асинхронного обмена в данном случае передачи прост.

1. Процессор микроЭВМ проверяет готовность ВУ к приему данных.

2. Если ВУ готово к приему данных (логический 0 в регистре *А2*), то данные передаются из шины данных системного интерфейса в регистр данных *А1* контроллера и далее в ВУ. Иначе повторяется п. 1.

Для написания программы асинхронной передачи воспользуемся командами микроЭВМ «Электроника-60» (см. приложение В).

Пример 8.1. Фрагмент программы передачи байта данных в асинхронном режиме с использованием параллельного интерфейса (контроллер ВУ, рис. 8.6):

M1: TST @# A2 – проверка содержимого регистра *А2*, т. е. готовности ВУ к приему данных. При выполнении команды процессор по шине адреса передает в контроллер адрес *А2*, сопровождая его сигналом «Ввод». Логика управления контроллера, реагируя на эти сигналы, обеспечивает передачу в процессор содержимого регистра состояния и управления *А2* по одной из линий шины данных системного интерфейса (например, это может быть старший — 15-й разряд шины данных).

BMI M1 — переход, если минус. Процессор проверяет значение старшего (знакового) разряда принятого слова данных. Единица в старшем разряде (знак минус) указывает на неготовность ВУ к приему данных и, следовательно, на необходимость возврата к проверке содержимого *А2*, т. е. процессор, выполняя две первые команды, ожидает готовности ВУ к при-

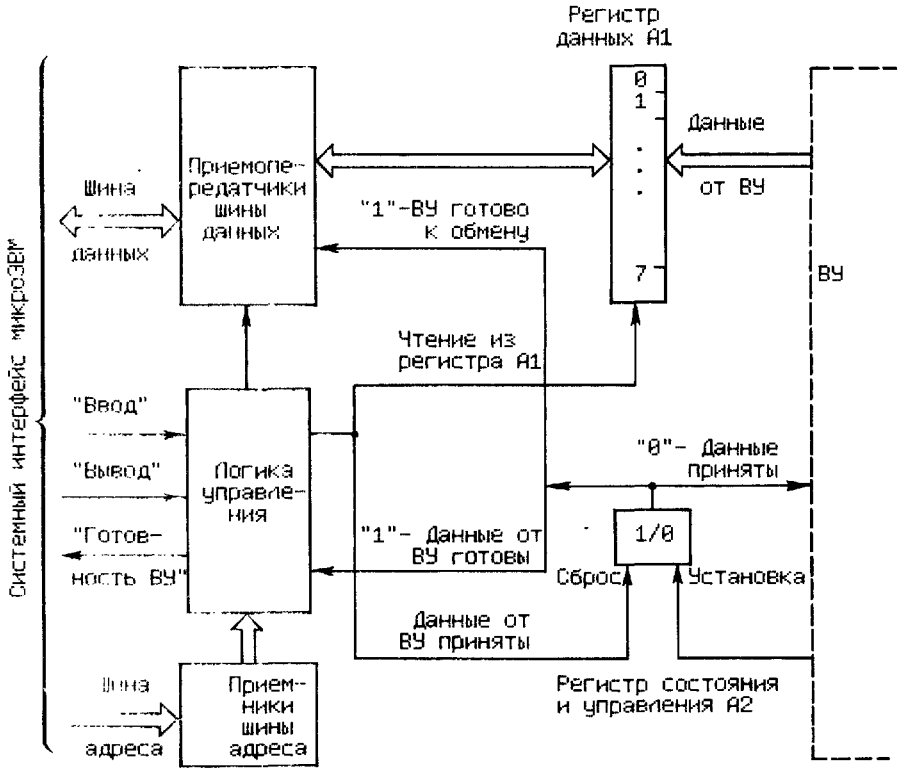


Рис. 8.7. Простой параллельный интерфейс (контроллер) ввода

ему данных. Нуль в старшем разряде подтверждает готовность ВУ и, следовательно, возможность передачи байта данных.

MOV R1, @ # A1 — пересылка данных из регистра R1 процессора в регистр данных контроллера *A1*. Процессор по шине адреса передает в контроллер адрес *A1*, а по шине данных — байт данных, сопровождая их сигналом «Вывод». Логика управления контроллера обеспечивает запись байта данных с шины данных в регистр данных *A1* и устанавливает в единицу регистр состояния и управления *A2*, формируя тем самым управляющий сигнал для ВУ «Выходные данные готовы». ВУ принимает байт данных и управляющим сигналом «Данные приняты» обнуляет регистр состояния и управления *A2*. Контроллер ВУ по этому сигналу формирует и передает в процессор сигнал «Готовность ВУ», который в данном случае извещает процессор о приеме байта данных внешним устройством и завершает цикл вывода данных в команде пересылки (см. также рис. 8.2, *а*).

Блок-схема простого контроллера ВУ (параллельного интерфейса), обеспечивающего побайтный прием данных из ВУ, приведена на рис. 8.7. В этом контроллере при взаимодействии с внешним устройством также используются два управляющих сигнала «Данные от ВУ готовы» и «Данные приняты».

Для формирования управляющего сигнала «Данные приняты» и приема из ВУ управляющего сигнала «Данные от ВУ готовы» используется одноразрядный адресуемый регистр состояния и управления *A2*.

ВУ записывает в регистр данных контроллера *A1* очередной байт данных и управляющим сигналом «Данные от ВУ готовы» устанавливает в единицу регистр состояния и управления *A2*.

При этом формируются:

управляющий сигнал системного интерфейса «Готовность ВУ»;

признак готовности ВУ к обмену, передаваемый в процессор по одной из линий шины данных системного интерфейса посредством операции ввода при реализации программы асинхронного обмена.

Тем самым контроллер извещает процессор о готовности данных в регистре *A1*. Процессор, выполняя программу асинхронного обмена, читает байт данных из регистра данных контроллера и обнуляет регистр состояния и управления *A2*. При этом формируется управляющий сигнал «Данные приняты» в шине связи с ВУ.

Логика управления контроллера и приемопередатчики шин системного интерфейса выполняют те же функции, что и в контроллере вывода (см. рис. 8.6).

Рассмотрим работу параллельного интерфейса ввода при реализации программы асинхронного обмена. Алгоритм асинхронного ввода так же прост, как и асинхронного вывода.

1. Процессор проверяет наличие данных в регистре данных контроллера *A1*.

2. Если данные готовы (логическая 1 в регистре *A2*), то они передаются из регистра данных *A1* в шину данных системного интерфейса и далее в регистр процессора или ячейку памяти микроЭВМ. Иначе повторяется п. 1.

Пример 8.2. Фрагмент программы приема байта данных в асинхронном режиме с использованием параллельного интерфейса (контроллер ВУ, рис. 8.7):

M1: TST @ # A2 — проверка содержимого регистра *A2* т. е. признака наличия данных в регистре данных *A1*. Команда выполняется точно так же, как и первая команда в примере 8.1.

BPL M1 — переход, если плюс. Процессор проверяет значение старшего (знакового) разряда принятого слова данных. Единица в знаковом разряде (содержимое регистра *A2*) подтверждает, что данные от ВУ записаны в регистр данных контроллера и необходимо переслать их в шину данных. Нуль в знаковом разряде указывает на неготовность данных от ВУ и, следовательно, на необходимость вернуться к проверке готовности (команда TST @ # A2).

MOV @ # A1,R1 — пересылка данных из регистра данных контроллера *A1* в регистр R1 процессора. Процессор передает в контроллер по шине адреса системного интерфейса адрес *A1*, сопровождая его сигналом «Ввод». Логика управления контроллера в ответ на сигнал «Ввод» обеспечивает передачу байта данных из регистра данных *A1* на шину данных, сопровождая их сигналом «Готовность ВУ», который подтверждает наличие данных от ВУ на шине данных (см. также рис. 8.2, *б*). Затем логика управления обнуляет регистр состояния и управления *A2*, формируя тем самым управляющий сигнал для ВУ «Данные приняты». Таким образом завершается цикл ввода данных в команде пересылки.

Как видно из рассмотренных примеров, для приема или передачи одного байта данных процессору необходимо выполнить всего несколько команд, время осуществления которых и определяет максимально достижимую скорость обмена данными при параллельной передаче. При этом время обработки байта данных в электронных схемах контроллера можно не учитывать, так как оно существенно меньше времени выполнения одной команды процессора. Таким образом, при параллельной передаче обеспечивается довольно высокая скорость обмена данными. Зачастую она достигает сотен килобайтов в секунду и на практике ограничивается только быстродействием ВУ.

Простота реализации и высокая скорость обмена, присущие параллельной передаче данных, обеспечили широкое распространение этого способа обмена данными с внешними устройствами в ЭВМ. Современные микроЭВМ оснащают широким набором универсальных параллельных интерфейсов. Примером могут служить параллельные интерфейсы типов И2, И4, ... для микроЭВМ «Электроника-60». Параллельные интерфейсы выпускают и в виде отдельных БИС. В гл. 10 будет рассмотрена БИС КР580ВВ55 — программируемый параллельный интерфейс.

8.4. Последовательная передача данных

Использование последовательных линий связи для обмена данными с ВУ возлагает на контроллеры ВУ дополнительные по сравнению с контроллерами для параллельного обмена функции. Во-первых, возникает необходимость преобразования формата данных: из параллельного формата, в котором они поступают в контроллер ВУ из системного интерфейса микро-ЭВМ, в последовательный при передаче в ВУ и из последовательного в параллельный при приеме данных из ВУ. Во-вторых, требуется реализовать соответствующий режиму работы ВУ способ обмена данными: синхронный или асинхронный.

Синхронные последовательные интерфейсы. Простой контроллер для синхронной передачи данных в ВУ по последовательной линии связи (последовательный интерфейс) представлен на рис. 8.8.

Восьмиразрядный адресуемый буферный регистр контроллера *A1* служит для временного хранения байта данных до его загрузки в сдвиговый регистр. Запись байта данных в буферный регистр из шины данных системного интерфейса производится так же, как и в параллельном интерфейсе (см. параграф 8.3 и рис. 8.6), только при наличии единицы в одноразрядном адресуемом регистре состояния контроллера *A2*. Единица в регистре состояния указывает на готовность контроллера принять очередной байт в буферный регистр. Содержимое регистра *A2* передается в процессор по одной из линий шины данных системного интерфейса и используется для формирования управляющего сигнала системного интерфейса «Готовность ВУ». При записи очередного байта в буферный регистр *A1* обнуляется регистр состояния *A2*.

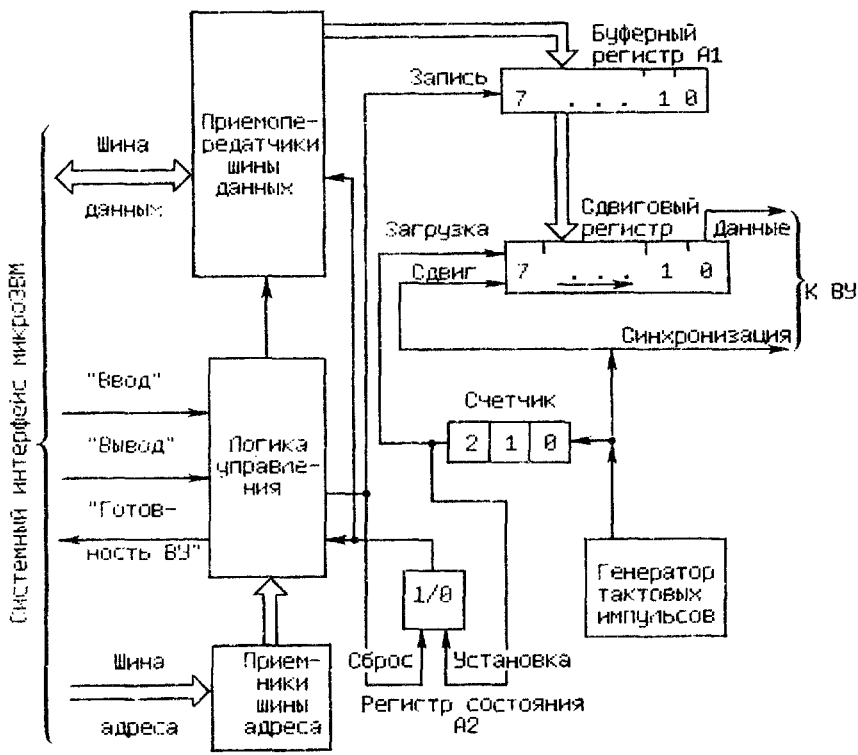


Рис. 8.8. Контроллер последовательной синхронной передачи

Программа записи байта данных в буферный регистр аналогична программе из примера 8.1 за исключением команды перехода: вместо команды BMI M1 (переход, если минус) нужно использовать команду BPL M1 (переход, если плюс).

Преобразование данных из параллельного формата, в котором они поступили в буферный регистр контроллера из системного интерфейса, в последовательный и передача их в линию связи производятся в сдвиговом регистре с помощью генератора тактовых импульсов и двоичного трехразрядного счетчика импульсов следующим образом.

Последовательная линия связи контроллера с ВУ подключается к выходу младшего разряда сдвигового регистра. По очередному тактовому импульсу содержимое сдвигового регистра сдвигается на один разряд вправо и в линию связи «Данные» выдается значение очередного разряда. Одновременно со сдвигом в ВУ передается по отдельной линии «Синхронизация» тактовый импульс. Таким образом, каждый передаваемый по линии «Данные» бит информации сопровождается синхронизирующим сигналом по линии «Синхронизация», что обеспечивает его однозначное восприятие на приемном конце последовательной линии связи.

Количество переданных в линию тактовых сигналов, а следовательно, и переданных бит информации подсчитывается счетчиком тактовых импульсов. Как только содержимое счетчика становится равным 7, т. е. в линию переданы 8 бит (1 байт) информации, формируется управляющий сигнал «Загрузка», обеспечивающий запись в сдвиговый регистр очередного байта из буферного регистра. Этим же управляющим сигналом устанавливается в «1» регистр состояния. Очередным тактовым импульсом счетчик будет сброшен в «0», и начнется очередной цикл выдачи восьми битов информации из сдвигового регистра в линию связи.

Синхронная последовательная передача отдельных битов данных в линию связи должна производиться без какого-либо перерыва, и следующий байт данных должен быть загружен в буферный регистр из системного интерфейса за время, не превышающее времени передачи восьми битов в последовательную линию связи.

При записи байта данных в буферный регистр обнуляется регистр состояния контроллера. Нуль в этом регистре указывает, что в линию связи передается байт данных из сдвигового регистра, а следующий передаваемый байт данных загружен в сдвиговый регистр.

Простой контроллер для последовательного синхронного приема данных из ВУ (рис. 8.9) состоит из тех же компонентов, что и контроллер для синхронной последовательной передачи, за исключением генератора тактовых импульсов.

Буферный регистр контроллера *A1* служит для временного хранения байта данных, поступившего из сдвигового регистра. Чтение байта данных в системный интерфейс из буферного регистра производится так же, как и в параллельном интерфейсе (см. параграф 8.3 и рис. 8.7). Единица в регистре состояния контроллера *A2* указывает на готовность контроллера передать очередной байт данных в системный интерфейс. Программа чтения байта данных из буферного регистра контроллера ВУ в процессор микроЭВМ аналогична программе из примера 8.2.

Данные, поступающие из линии связи в последовательном коде, преобразуются в контроллере в параллельный код с помощью сдвигового регистра и трехразрядного двоичного счетчика тактовых импульсов следующим образом.

Входная последовательная линия связи «Данные» подключается в контроллере к последовательному входу сдвигового регистра, а входная линия

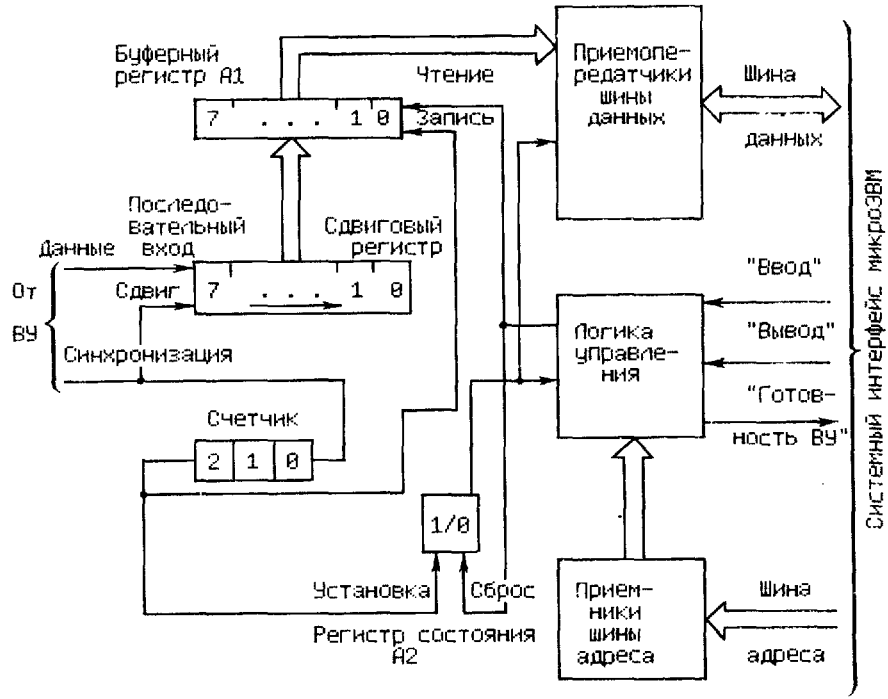


Рис. 8.9. Контроллер последовательного синхронного приема

«Синхронизация» – на управляющий вход «Сдвиг» сдвигового регистра и на вход счетчика тактовых импульсов. По очередному тактовому сигналу, поступившему от синхрогенератора ВУ по линии «Синхронизация», производятся сдвиг содержимого сдвигового регистра на один разряд влево и запись очередного бита данных из линии связи «Данные» в младший разряд этого регистра. Одновременно увеличивается на единицу содержимое счетчика тактовых импульсов. Как только содержимое счетчика становится равным 7, т. е. в сдвиговый регистр приняты последовательно восемь битов информации, формируется управляющий сигнал «Запись», который обеспечивает запись в буферный регистр очередного принятого байта из сдвигового регистра. Этим же управляющим сигналом устанавливается в «1» регистр состояния.

За время приема в сдвиговый регистр следующих восьми битов информации байт данных из буферного регистра должен быть передан в шину данных системного интерфейса микроЭВМ. При этой передаче обнуляется регистр состояния контроллера и нуль в этом регистре означает, что в сдвиговый регистр принимается из линии связи очередной байт информации.

Асинхронные последовательные интерфейсы. Организация асинхронного последовательного обмена данными с ВУ осложняется тем, что на передающей и приемной стороне последовательной линии связи используются настроенные на одну частоту, но физически разные генераторы тактовых импульсов и, следовательно, общая синхронизация отсутствует. Рассмотрим на примерах организацию контроллеров последовательных интерфейсов для последовательных асинхронных передачи и приема данных.

Простейший контроллер для асинхронной передачи данных в ВУ по последовательной линии связи представлен на рис. 8.10. Он предназначен для передачи данных в формате с двумя стоповыми битами.

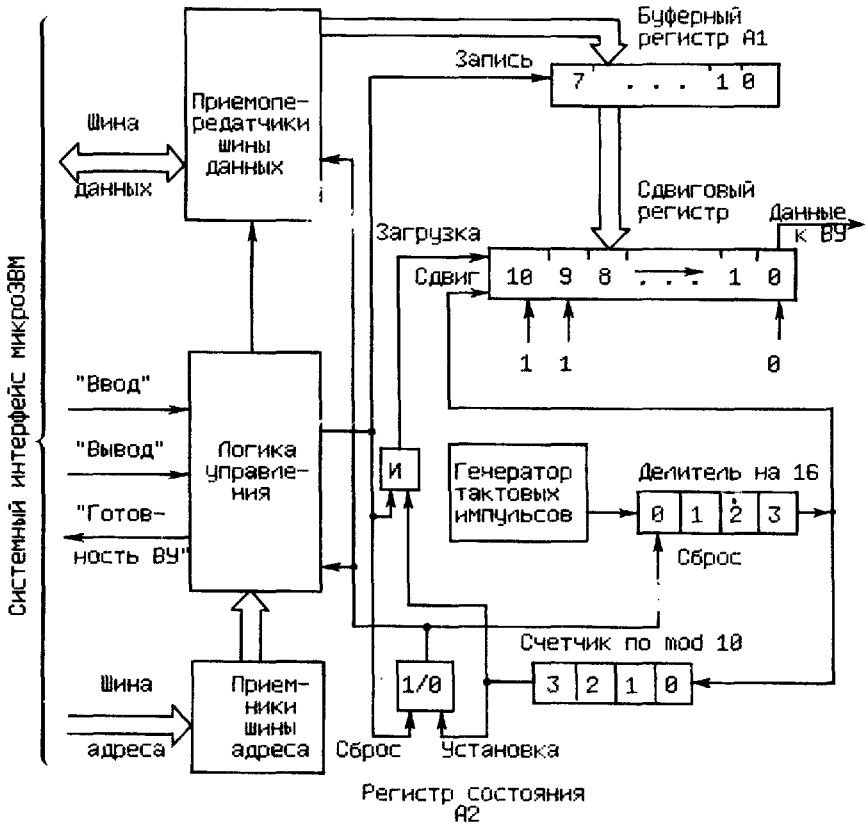


Рис. 8.10. Контроллер последовательной асинхронной передачи

После передачи очередного байта данных в регистр состояния *A2* записывается 1. Единичный выходной сигнал регистра *A2* информирует процессор о готовности контроллера к приему следующего байта данных и передаче его по линии связи в ВУ. Этот же сигнал запрещает формирование импульсов со схемы выработки импульсов сдвига — делителя частоты сигналов тактового генератора на 16. Счетчик импульсов сдвига (счетчик по mod 10) находится в нулевом состоянии, и его единичный выходной сигнал поступает на вентиль *И*, подготавливая цепь выработки сигнала загрузки сдвигового регистра.

Процесс передачи байта данных начинается с того, что процессор, выполняя команду «Вывод» (см. пример 8.1), выставляет этот байт на шине данных. Одновременно процессор формирует управляющий сигнал системного интерфейса «Вывод», по которому производятся запись передаваемого байта в буферный регистр *A1*, сброс регистра состояния *A2* и формирование на вентиле *И* сигнала «Загрузка». Передаваемый байт переписывается в разряды 1, ... , 8 сдвигового регистра, в нулевой разряд сдвигового регистра записывается 0 (стартовый бит), а в разряды 9 и 10 — 1 (стоповые биты). Кроме того, снимается сигнал «Сброс» с делителя частоты, он начинает накапливать импульсы генератора тактовой частоты и в момент приема шестнадцатого тактового импульса вырабатывает импульс сдвига.

На выходной линии контроллера «Данные» поддерживается состояние 0 (значение стартового бита) до тех пор, пока не будет выработан пер-

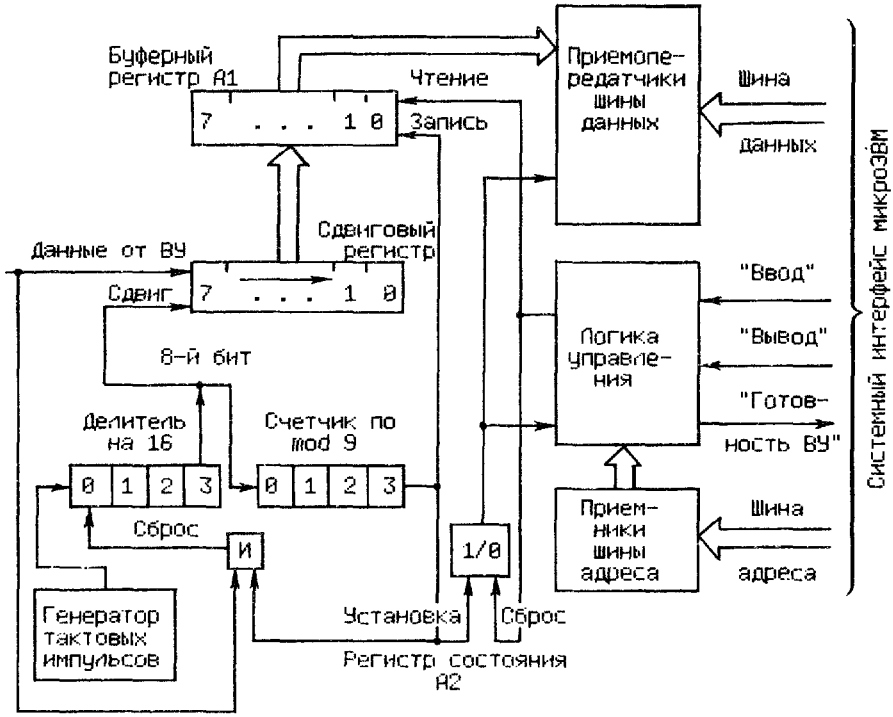


Рис. 8.11. Контроллер последовательного асинхронного приема

вый импульс сдвига. Импульс сдвига изменит состояние счетчика импульсов сдвига и перепишет в нулевой разряд сдвигового регистра первый информационный бит передаваемого байта данных. Состояние, соответствующее значению этого бита, будет поддерживаться на линии «Данные» до следующего импульса сдвига.

Аналогично будут переданы остальные информационные биты, первый стоповый бит и, наконец, второй стоповый бит, при передаче которого счетчик импульсов сдвига снова установится в нулевое состояние. Это приведет к записи 1 в регистр состояния *A2*. Единичный сигнал с выхода регистра *A2* запретит формирование импульсов сдвига, а также информирует процессор о готовности к приему нового байта данных. После завершения передачи очередного кадра (стартового бита, информационного байта и двух стоповых битов) контроллер поддерживает в линии связи уровень логической единицы (значение второго стопового бита).

Уровень логической единицы поступает по линии «Данные» в контроллер для асинхронного приема данных (рис. 8.11). Этот уровень создает условия для выработки сигнала, запрещающего работу делителя частоты генератора тактовых импульсов. Действительно, после приема предыдущего байта данных счетчик импульсов сдвига (счетчик по mod 9) находится в нулевом состоянии и на вентиль *И* поступают два единичных сигнала: со счетчика сдвигов и из линии «Данные». На выходе вентиля *И* вырабатывается сигнал сброса делителя частоты сигналов тактового генератора, запрещающий формирование импульсов сдвига.

В момент смены стопового бита на стартовый бит (момент начала передачи нового кадра) на линии «Данные» появится уровень логического нуля и тем самым снят сигнал сброса с делителя частоты. Состояние 4-разряд-

ного двоичного счетчика (делителя частоты) начнет изменяться. Когда на счетчике накопится значение 8, он выдаст сигнал, поступающий на входы сдвигового регистра и счетчика импульсов сдвига. Так как частота сигналов генератора тактовых импульсов приемника должна совпадать с частотой генератора тактовых импульсов передатчика, то сдвиг (считывание) бита произойдет примерно на середине временнóго интервала, отведенного на передачу бита данных, т. е. времени, необходимого для выработки шестнадцати тактовых импульсов. Это делается для уменьшения вероятности ошибки из-за возможного различия частот генераторов передатчика и приемника, искажения формы передаваемых сигналов (переходные процессы) и т. п. Следующий сдвиг произойдет после прохождения шестнадцати тактовых импульсов, т. е. на середине временнóго интервала передачи первого информационного бита.

При приеме в сдвиговый регистр девятого бита кадра (восьмого информационного бита) из него «выдвинется» стартовый бит и, следовательно, в сдвиговом регистре будет размещен весь принятый байт информации. В этот момент счетчик импульсов сдвига придет в нулевое состояние и на его выходе будет выработан единичный сигнал, по которому:

содержимое сдвигового регистра перепишется в буферный регистр;

в регистр состояния *А2* запишется 1 и он будет информировать процессор об окончании приема очередного байта;

вентиль *И* подготовится к выработке сигнала «Сброс» (этот сигнал сформируется после прихода первого стопового бита).

Получив сигнал готовности (1 в регистре *А2*), процессор выполнит команду «Ввод» (см. пример 8.2). При этом вырабатывается управляющий сигнал системного интерфейса «Ввод», по которому производятся пересылка принятого байта данных из буферного регистра в процессор (сигнал «Чтение») и сброс регистра состояния *А2*.

Отметим, что для простоты изложения в контроллере на рис. 8.11 не показаны схемы контроля стоповых битов принимаемого кадра. Не показаны также схемы контроля четности или нечетности (паритета) передаваемой информации (обычно в передаваемом байте восьмому биту придается значение 0 или 1 так, чтобы в этом байте было четное количество единиц). В реальных контроллерах имеются такие схемы, и если контроллер не принимает из линии связи нужного количества стоповых битов или вырабатывается сигнал ошибки паритета в схеме контроля четности, то принятые в текущем кадре биты данных игнорируются и контроллер ожидает поступления нового стартового бита.

Обмен данными с ВУ по последовательным линиям связи широко используется в современных микроЭВМ, особенно в тех случаях, когда не требуется высокой скорости обмена. Вместе с тем применение в микроЭВМ последовательных линий связи с ВУ обусловлено двумя важными причинами. Во-первых, последовательные линии связи просты по своей организации: два провода при симплексной и полудуплексной передаче или четыре — при дуплексной. Во-вторых, в микроЭВМ широко используются ВУ, обмен с которыми необходимо вести в последовательном коде.

В современных микроЭВМ применяют, как правило, универсальные контроллеры для последовательного обмена, т. е. обеспечивающие как синхронный, так и асинхронный режим обмена данными с ВУ. Примером такого контроллера может служить «Устройство последовательного обмена» из состава модулей микроЭВМ типа «Электроника-60». Большое распространение получили и БИС — универсальные приемопередатчики для последовательного обмена данными. Типичным примером такой БИС является БИС КР580ВВ51, работа которой будет рассмотрена в гл. 10.

8.5. Организация прерываний в микроЭВМ

Одной из разновидностей программно-управляемого обмена данными с ВУ в микроЭВМ, как отмечалось в гл. 3, является **обмен с прерыванием программы** отличающийся от асинхронного программно-управляемого обмена тем, что переход к выполнению команд, физически реализующих обмен данными, осуществляется с помощью специальных аппаратных средств. Команды обмена данными в этом случае выделяют в отдельный программный модуль - подпрограмму обработки прерывания. Задачей аппаратных средств обработки прерывания в процессоре микроЭВМ как раз и является приостановка выполнения одной программы (ее еще называют основной программой) и передача управления подпрограмме обработки прерывания. Действия, выполняемые при этом процессором, как правило, те же, что и при обращении к подпрограмме. Только при обращении к подпрограмме они инициируются командой, а при обработке прерывания — управляющим сигналом от ВУ, который называют **«Требование прерывания»** или **«Запрос на прерывание»**.

Эта важная особенность обмена с прерыванием программы позволяет организовать обмен данными с ВУ в произвольные моменты времени, не зависящие от программы, выполняемой в микроЭВМ. Таким образом, появляется возможность обмена данными с ВУ в реальном масштабе времени, определяемом внешней по отношению к микроЭВМ средой. Обмен с прерыванием программы существенным образом экономит время процессора, затрачиваемое на обмен. Это происходит за счет того, что исчезает необходимость в организации программных циклов ожидания готовности ВУ (см. примеры 8.1 и 8.2), на выполнение которых тратится значительное время, особенно при обмене с медленнодействующими ВУ.

Прерывание программы по требованию ВУ не должно оказывать на прерванную программу никакого влияния кроме увеличения времени ее выполнения за счет приостановки на время выполнения подпрограммы обработки прерывания. Поскольку для выполнения подпрограммы обработки прерывания используются различные регистры процессора (счетчик команд, регистр состояния и т. д.), то информацию, содержащуюся в них в момент прерывания, необходимо сохранить для последующего возврата в прерванную программу.

Обычно задача сохранения содержимого счетчика команд и регистра состояния процессора возлагается на аппаратные средства обработки прерываний. Сохранение содержимого других регистров процессора, используемых в подпрограмме обработки прерывания, производится непосредственно в подпрограмме. Отсюда следует достаточно очевидный факт: чем больший объем информации о прерванной программе сохраняется программным путем, тем больше время реакции микроЭВМ на сигнал прерывания, и наоборот. Предпочтительными с позиции повышения производительности микроЭВМ (сокращения времени выполнения подпрограмм обработки, а следовательно, и основной программы) являются уменьшение числа команд, обеспечивающих сохранение информации о прерванной программе, и реализация этих функций аппаратными средствами.

Формирование сигналов прерываний — запросов ВУ на обслуживание - происходит в контроллерах соответствующих ВУ. В простейших случаях в качестве сигнала прерывания может использоваться сигнал «Готовность ВУ» (см. рис. 8.4, *а*, *б*), поступающий из контроллера ВУ в системный интерфейс микроЭВМ. Однако такое простое решение обладает существенным недостатком - процессор не имеет возможности управлять прерываниями, т. е. разрешать или запрещать их для отдельных ВУ. В результате

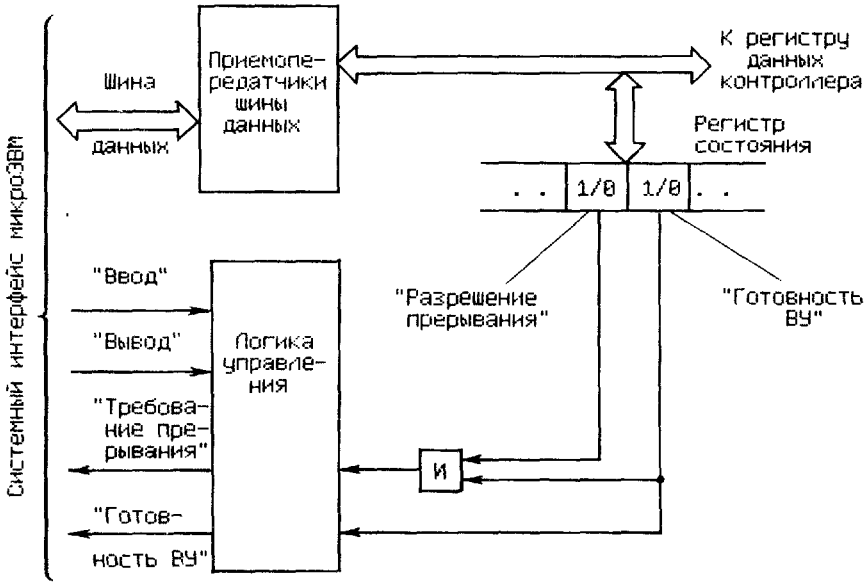


Рис. 8.12. Фрагмент блок схемы контроллера ВУ с разрядом «Разрешение прерывания» в регистре состояния

организация обмена данными в режиме прерывания с несколькими ВУ существенно усложняется.

Для решения этой проблемы регистр состояния контроллера ВУ (рис. 8.12) дополняют еще одним разрядом «Разрешение прерывания». Запись 1 или 0 в разряд «Разрешение прерывания» регистра состояния производится программным путем по одной из линий шины данных системного интерфейса. Управляющий сигнал системного интерфейса «Требование прерывания» формируется с помощью схемы совпадения только при наличии единиц в разрядах «Готовность ВУ» и «Разрешение прерывания» регистра состояния контроллера.

Аналогичным путем решается проблема управления прерываниями в микроЭВМ в целом. Для этого в регистре состояния процессора выделяется разряд, содержимое которого определяет, разрешены или запрещены прерывания от внешних устройств. Значение этого разряда может устанавливаться программным путем, как это было реализовано в базовой ЭВМ.

В микроЭВМ обычно используется одноуровневая система прерываний, т. е. сигналы «Требование прерывания» от всех ВУ поступают на один вход процессора. Поэтому возникает проблема идентификации ВУ, запросившего обслуживание, и реализации заданной очередности (приоритета) обслуживания ВУ при одновременном поступлении нескольких сигналов прерывания. Существуют два основных способа идентификации ВУ, запросивших обслуживания:

программный опрос регистров состояния (разряд «Готовность ВУ») контроллеров всех ВУ;

использование векторов прерывания.

Организация прерываний с программным опросом готовности предполагает наличие в памяти микроЭВМ единой подпрограммы обслуживания прерываний от всех ВУ. Структура такой подпрограммы приведена на рис. 8.13. Обслуживание ВУ с помощью единой подпрограммы обработки прерываний производится следующим образом.

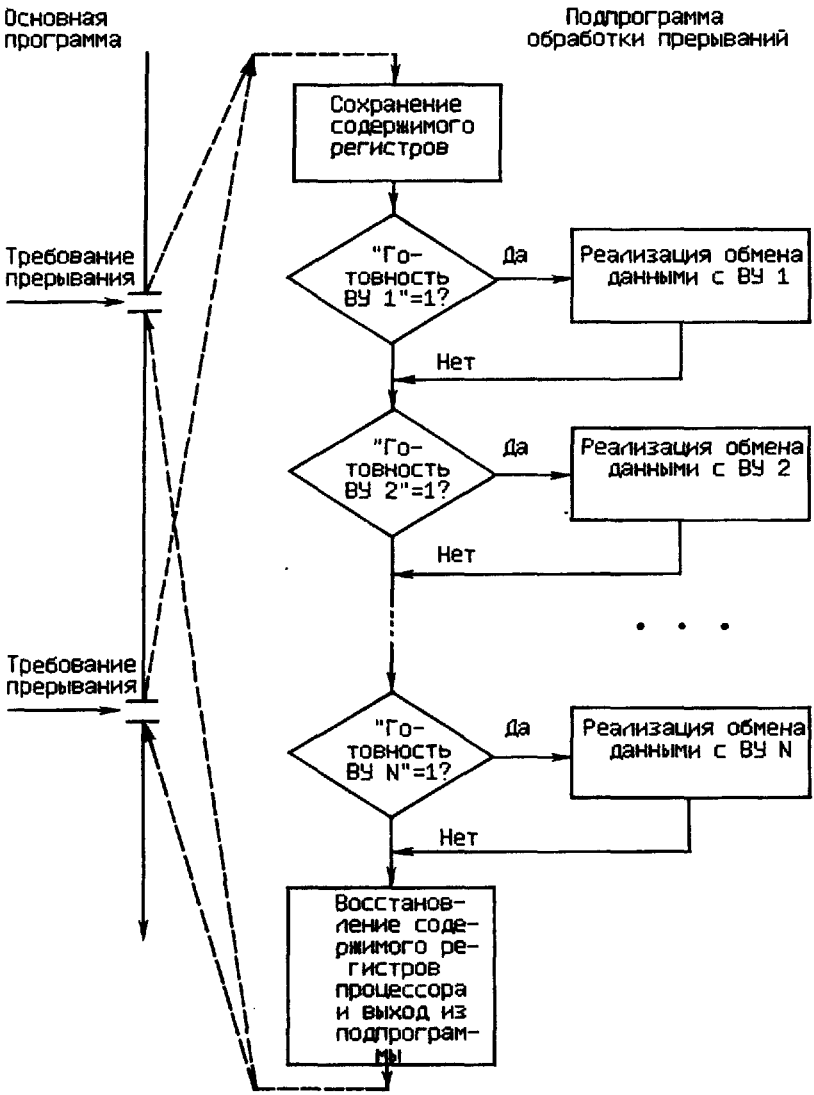


Рис. 8.13. Структура подпрограммы обработки прерывания и ее связь с основной программой

В конце последнего машинного цикла выполнения очередной команды основной программы процессор проверяет наличие требования прерывания от ВУ. Если сигнал прерывания есть и в процессоре прерывание разрешено, то процессор переключается на выполнение подпрограммы обработки прерываний.

После сохранения содержимого регистров процессора, используемых в подпрограмме, начинается последовательный опрос регистров состояния контроллеров всех ВУ, работающих в режиме прерывания. Как только подпрограмма обнаружит готовое к обмену ВУ, сразу выполняются действия по его обслуживанию. Завершается подпрограмма обработки прерывания после опроса готовности всех ВУ и восстановления содержимого регистров процессора.

Приоритет ВУ в микроЭВМ с программным опросом готовности ВУ однозначно определяется порядком их опроса в подпрограмме обработки прерываний. Чем раньше в подпрограмме опрашивается готовность ВУ, тем меньше время реакции на его запрос и выше приоритет. Необходимость проверки готовности всех ВУ существенно увеличивает время обслуживания тех ВУ, которые опрашиваются последними. Это является основным недостатком рассматриваемого способа организации прерываний. Поэтому обслуживание прерываний с опросом готовности ВУ используется только в тех случаях, когда отсутствуют жесткие требования на время обработки сигналов прерывания ВУ.

Организация системы прерываний микроЭВМ с использованием векторов прерываний позволяет устранить указанный недостаток. При такой организации системы прерываний ВУ, запросившее обслуживания, само идентифицирует себя с помощью вектора прерывания — адреса ячейки основной памяти микроЭВМ, в которой хранится либо первая команда подпрограммы обслуживания прерывания данного ВУ, либо адрес начала такой подпрограммы. Таким образом, процессор, получив из контроллера ВУ вектор прерывания, сразу переключается на выполнение требуемой подпрограммы обработки прерывания. В микроЭВМ с векторной системой прерывания каждое ВУ должно иметь собственную подпрограмму обработки прерывания.

Вектор прерывания выдается контроллером не одновременно с запросом на прерывание, а только по разрешению процессора, как это реализовано в схеме на рис. 8.14. Это делается для того, чтобы исключить одновременную выдачу векторов прерывания от нескольких ВУ. В ответ на сигнал котроллера ВУ «Требование прерывания» процессор формирует управляющий сигнал «Предоставление прерывания (вх.)», который разрешает контроллеру ВУ, запросившему обслуживание, выдачу вектора прерывания в шину адреса системного интерфейса. Для этого в контроллере используются регистр вектора прерывания и схема совпадения И3. Регистр вектора прерывания обычно реализуется с помощью перемычек или переключателей, что позволяет пользователю микроЭВМ устанавливать для конкретных ВУ требуемые значения векторов прерывания.

Управляющий сигнал «Предоставление прерывания (вых.)» формируется в контроллере ВУ с помощью схемы совпадения И2. Этот сигнал используется для организации последовательного аппаратного опроса готовности ВУ и реализации тем самым требуемых приоритетов ВУ. Процессор при поступлении в него по общей линии системного интерфейса «Требование прерывания» сигнала прерывания формирует управляющий сигнал «Предоставление прерывания (вх.)», который поступает сначала в контроллер ВУ с наивысшим приоритетом (рис. 8.15). Если это устройство не требовало обслуживания, то его контроллер пропускает сигнал «Предоставление прерывания» на следующий контроллер, иначе дальнейшее распространение сигнала прекращается и контроллер выдает в шину адреса вектор прерывания.

Аппаратный опрос готовности ВУ производится гораздо быстрее, нежели программный. Но если обслуживания запросили одновременно два или более ВУ, обслуживание менее приоритетных ВУ будет отложено на время обслуживания более приоритетных, как и в системе прерывания с программным опросом.

Рассмотренная векторная система прерываний практически полностью соответствует системе прерываний, реализованной в микроЭВМ «Электроника-60». Восьмиразрядный вектор прерывания в «Электронике-60» указывает одну из ячеек памяти с адресами от 0 до $(376)_8$, в которой размещается

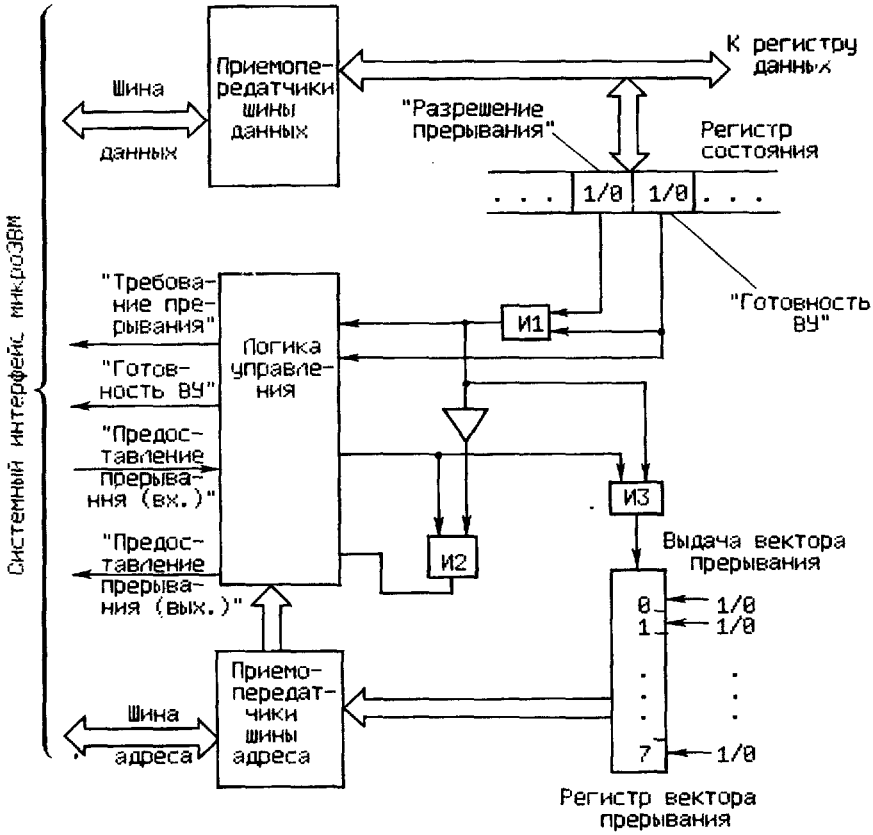


Рис. 8.14. Формирование вектора прерывания в контроллере ВУ

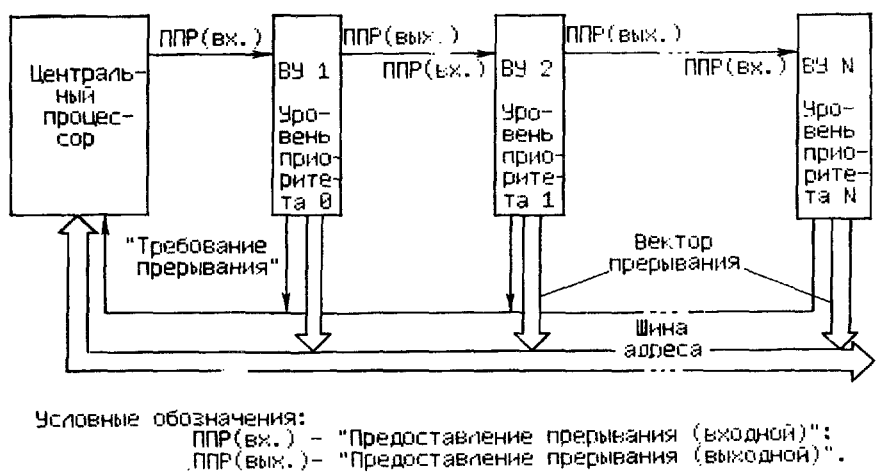


Рис. 8.15. Реализация приоритетов ВУ в микроЭВМ с векторной системой прерывания

адрес начала подпрограммы обработки прерывания. В следующей за указанной вектором прерывания ячейке памяти хранится новое содержимое регистра состояния процессора, загружаемое в него при переключении на подпрограмму обработки прерывания. Один из битов нового содержимого регистра состояния процессора запрещает или разрешает прерывания от других ВУ, что позволяет ВУ с более высоким приоритетом прерывать подпрограммы обслуживания ВУ с меньшим приоритетом и наоборот.

Для организации векторной системы прерываний в микроЭВМ могут использоваться специальные БИС, например БИС программируемого контроллера прерываний КР580ВН59, работа которой рассмотрена в гл. 10.

8.6. Организация прямого доступа к памяти

Одним из способов обмена данными с ВУ является обмен в режиме прямого доступа к памяти (ПДП). В этом режиме обмен данными между ВУ и основной памятью микроЭВМ происходит без участия процессора. Обменом в режиме ПДП управляет не программа, выполняемая процессором, а электронные схемы, внешние по отношению к процессору. Обычно схемы, управляющие обменом в режиме ПДП, размещаются в специальном контроллере, который называется контроллером прямого доступа к памяти.

Необходимость реализации в микроЭВМ обмена данными в режиме ПДП вызывается двумя основными факторами. Во-первых, при использовании режима ПДП появляется возможность начальной загрузки программ в основную память микроЭВМ из устройства ввода. Во-вторых, что и является наиболее важным, режим ПДП обеспечивает возможность использования в микроЭВМ быстродействующих внешних запоминающих устройств, таких как накопители на магнитных лентах (НМЛ) и магнитных дисках (НМД).

Это связано с тем, что обмен данными с ВЗУ (НМД, НМЛ) производится блоками фиксированного размера от 128 байт и более. Причем обмен должен осуществляться в строгой последовательности, соответствующей расположению информации на магнитном носителе, без каких-либо пропусков. Пропуск хотя бы одного байта вызывает необходимость в повторном обмене. При этом время, отводимое на обмен одним байтом данных, жестко ограничивается скоростью перемещения магнитного носителя относительно магнитных головок и не превышает, как правило, нескольких микросекунд. Обеспечить обмен блоком данных с ВЗУ с помощью программно управляемого обмена не представляется возможным, так как на обмен каждым байтом затрачивается несколько команд процессора, суммарное время выполнения которых в современных микроЭВМ превышает максимально допустимое время на обмен одним байтом с ВЗУ. Такой обмен обеспечивается прямым доступом в память микроЭВМ, при котором время на обмен одним байтом данных между памятью и ВЗУ равно циклу памяти (см. параграф 6.1).

Для реализации режима прямого доступа в память необходимо обеспечить непосредственную связь контроллера ПДП и памяти микроЭВМ. Для этой цели можно было бы использовать специально выделенные шины адреса и данных, связывающие контроллер ПДП с памятью микроЭВМ. Но такое решение нельзя признать оптимальным, так как это приведет к значительному усложнению микроЭВМ в целом, особенно при подключении нескольких ВЗУ. В целях сокращения количества линий в шинах микроЭВМ контроллер ПДП подключается к памяти посредством шин адреса и данных системного интерфейса. При этом возникает проблема совместного использования шин системного интерфейса процессором и конт-

роллером ПДП. Можно выделить два основных способа ее решения: реализация обмена в режиме ПДП с «захватом цикла» и в режиме ПДП с блокировкой процессора.

Существуют две разновидности прямого доступа в память с «захватом цикла». Наиболее простой способ организации ПДП состоит в том, что для обмена используются те машинные циклы процессора, в которых он не обменивается данными с памятью. В такие циклы контроллер ПДП может обмениваться данными с памятью, не мешая работе процессора. Однако возникает необходимость выделения таких циклов, чтобы не произошло временнóго перекрытия обмена ПДП с операциями обмена, инициируемыми процессором. В некоторых процессорах формируется специальный управляющий сигнал, указывающий циклы, в которых процессор не использует память. При использовании других процессоров для выделения таких циклов необходимо применение в контроллерах ПДП специальных селектирующих схем, что, конечно же, усложняет их конструкцию. Применение рассмотренного способа организации ПДП не снижает производительности микроЭВМ, но при этом обмен в режиме ПДП возможен только в случайные моменты времени одиночными байтами или словами.

Наиболее распространенным является ПДП с «захватом цикла» и принудительным отключением процессора от шин системного интерфейса. Для реализации такого режима ПДП системный интерфейс микроЭВМ дополняется двумя линиями для передачи управляющих сигналов «Требование прямого доступа к памяти» (ТПДП) и «Предоставление прямого доступа к памяти» (ППДП).

Управляющий сигнал ТПДП формируется контроллером прямого доступа к памяти. Процессор, получив этот сигнал, приостанавливает выполнение очередной команды, не дожидаясь ее завершения, выдает в системный интерфейс управляющий сигнал ППДП и отключается от шин системного интерфейса. С этого момента все шины системного интерфейса управляются контроллером ПДП. Контроллер ПДП, используя шины системного интерфейса, осуществляет обмен одним байтом или словом данных с памятью микроЭВМ и затем, сняв сигнал ТПДП, возвращает управление системным интерфейсом процессору. Как только контроллер ПДП будет готов к обмену следующим байтом, он вновь «захватывает» цикл процессора и т. д. В промежутках между сигналами ТПДП процессор продолжает выполнять команды программы. Тем самым выполнение программы замедляется, но в меньшей степени, чем при обмене в режиме прерывания.

Применение в микроЭВМ обмена данными с ВУ в режиме ПДП всегда требует предварительной подготовки, а именно: для каждого ВУ необходимо выделить область памяти, используемую при обмене, и указать ее размер, т. е. количество записываемых в память или читаемых из памяти байтов (слов) информации. Следовательно, контроллер ПДП должен обязательно иметь в своем составе регистр адреса и счетчик байтов (слов). Перед началом обмена с ВУ в режиме ПДП процессор должен выполнить программу загрузки. Эта программа обеспечивает запись в указанные регистры контроллера ПДП начального адреса выделенной ВУ памяти и ее размера в байтах или словах в зависимости от того, какими порциями информации ведется обмен. Сказанное не относится к начальной загрузке программ в память микроЭВМ в режиме ПДП. В этом случае содержимое регистра адреса и счетчика байтов слов устанавливается переключателями или перемычками непосредственно на плате контроллера.

Блок-схема простого контроллера ПДП, обеспечивающего ввод данных в память микроЭВМ по инициативе ВУ в режиме ПДП «Захват цикла», приведена на рис. 8.16.

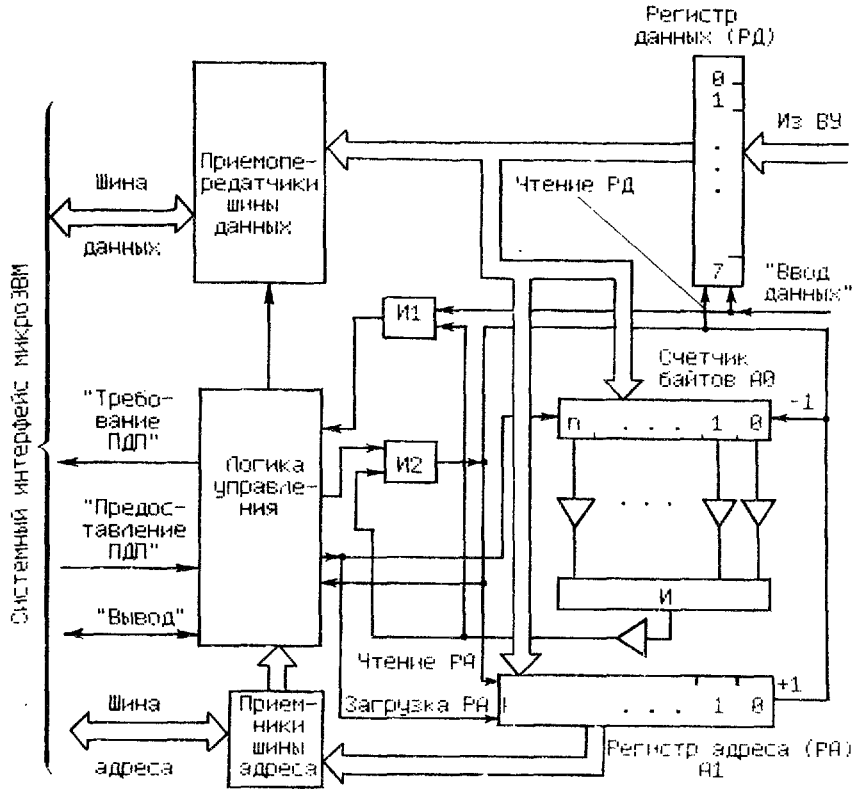


Рис. 8. 16. Контроллер ПДП для ввода данных из ВУ в режиме «Захват цикла»

Перед началом очередного сеанса ввода данных из ВУ процессор загружает в регистры его контроллера следующую информацию: в счетчик байтов — количество принимаемых байтов данных, а в регистр адреса — начальный адрес области памяти для вводимых данных. Тем самым контроллер подготавливается к выполнению операции ввода данных из ВУ в память микроЭВМ в режиме ПДП.

Байты данных из ВУ поступают в регистр данных контроллера в постоянном темпе. При этом каждый байт сопровождается управляющим сигналом из ВУ «Ввод данных», который обеспечивает запись байта данных в регистр данных контроллера. По этому же сигналу и при ненулевом состоянии счетчика байтов контроллер формирует сигнал ТПДП. По ответному сигналу процессора ППДП контроллер выставляет на шины адреса и данных системного интерфейса содержимое своих регистров адреса и данных соответственно. Формируя управляющий сигнал «Вывод», контроллер ПДП обеспечивает запись байта данных из своего регистра данных в память микроЭВМ. Сигнал ППДП используется в контроллере и для модификации счетчика байтов и регистра адреса. По каждому сигналу ППДП из содержимого счетчика байтов вычитается единица, и как только содержимое счетчика станет равно нулю, контроллер прекратит формирование сигналов «Требование прямого доступа к памяти».

На примере простого контроллера ПДП мы рассмотрели только процесс подготовки контроллера и непосредственно передачу данных в режиме ПДП. На практике любой сеанс обмена данными с ВУ в режиме ПДП

всегда инициируется программой, реализуемой процессором, и включает два следующих этапа.

1. На этапе подготовки ВУ к очередному сеансу обмена на процессор в режиме программно-управляемого обмена опрашивает состояние ВУ (проверяет его готовность к обмену) и посылает в ВУ команды, обеспечивающие его подготовку к обмену. Такая подготовка может сводиться, например, к перемотке магнитной ленты в НМЛ или перемещению головок на требуемую дорожку в НМД. Затем выполняется загрузка регистров контроллера ПДП. На этом подготовка к обмену в режиме ПДП завершается, и процессор переключается на выполнение другой программы.

2. Обмен данными в режиме ПДП начинается после завершения подготовительных операций в ВУ по инициативе либо ВУ, как это было рассмотрено выше, либо процессора. В этом случае контроллер ПДП необходимо дополнить регистром состояния и управления, содержимое которого будет определять режим работы контроллера ПДП. Один из разрядов этого регистра будет инициировать обмен данными с ВУ. Загрузка информации в регистр состояния и управления контроллера ПДП производится программным путем.

Здесь следует отметить, что использование в микроЭВМ обмена в режиме ПДП требует от программиста очень ясного понимания процессов, происходящих в микроЭВМ при выполнении программы, и четкой синхронизации процесса выполнения программы и ввода-вывода в режиме ПДП.

Прямой доступ в память с блокировкой процессора отличается от ПДП с «захватом цикла» тем, что управление системным интерфейсом передается контроллеру ПДП не на время обмена одним байтом, а на время обмена блоком данных. Такой режим ПДП необходим в тех случаях, когда время обмена одним байтом с ВУ сопоставимо с циклом процессора. В этом случае процессор не успевает выполнить хотя бы одну команду между очередными операциями обмена в режиме ПДП.

В микроЭВМ можно использовать несколько ВУ, работающих в режиме ПДП. Предоставление таким ВУ шин системного интерфейса для обмена данными производится на приоритетной основе. Приоритеты ВУ реализуются так же, как и при обмене данными в режиме прерывания, но вместо управляющих сигналов «Требование прерывания» и «Предоставление прерывания» (см. рис. 8.15) используются сигналы «Требование прямого доступа» и «Предоставление прямого доступа» соответственно.

Контроллеры ПДП реализуются и в виде отдельной БИС. В гл. 10 будет рассмотрена микросхема КР580ВТ57, управляющая работой четырех независимых каналов ПДП с учетом приоритетов подключенных к ним ВУ.

Глава 9

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ ФУНКЦИОНИРОВАНИЯ МИКРОЭВМ

9.1. Модельный подход при изучении устройств микропроцессорной техники

Глубокое понимание принципов построения и функционирования сложных систем (каковыми являются устройства микропроцессорной техники) достигается лишь при практической работе с такими системами или их моде-

лями. Даже самое подробное описание работы с базовой ЭВМ (см. параграф 3.1) и детальные иллюстрации процесса выполнения команд (см. например, рис. 2.2) не могут заменить действительной работы на этой ЭВМ, снабженной специальными пультами управления и индикации (см., например, рис. 3.1 и 4.7).

Казалось бы, при сравнительной дешевизне и широком распространении устройств микропроцессорной техники достаточно легко создать стенды для изучения принципов функционирования этих средств вычислительной и управляющей техники. Однако отсутствие доступа к большинству компонентов микропроцессорных устройств (регистров, сумматоров, вентильных схем и т. п.), расположенных внутри одной или нескольких БИС, приводит к необходимости использования для этих целей специально разработанных моделей: физических или функциональных.

Физическая модель — это стенд со структурой изучаемого устройства микропроцессорной техники, построенный на интегральных схемах малой и средней степени интеграции и снабженный средствами для управления режимами работы исследуемого устройства, а также индикации состояний любых его компонентов. В связи с быстрой эволюцией средств микропроцессорной техники массовый выпуск таких специализированных стендов нецелесообразен. Экономически невыгодно и их мелкосерийное производство.

Функциональная модель — это микроЭВМ, в которую введена программа, имитирующая работу исследуемого устройства микропроцессорной техники под воздействием приказов, подаваемых со специально разработанного пульта управления или, например, со штатной клавиатуры микроЭВМ. Состояния компонентов исследуемого устройства отображаются на индикаторах специального пульта или на дисплее микроЭВМ. Таким образом, в качестве технического средства для функционального моделирования цифровых устройств можно использовать персональную ЭВМ, например бытовой компьютер «Электроника БК-0010» — небольшой ящичек с клавиатурой (370×180×70 мм), подсоединяемый к бытовому телевизору (черно-белому или цветному) и бытовому кассетному магнитофону [4].

Если существует программа моделирования какого-либо цифрового устройства, составленная на языке персональной ЭВМ «Электроника БК-0010» и записанная на кассетной магнитной ленте, то после ввода такой ленты в машину на экране телевизора будет вычерчена структура исследуемого устройства, например структура контроллера асинхронного последовательного вывода (рис. 9.1). Кроме того, на экране будет воспроизведено (в виде символов 0 и 1) состояние регистров и счетчиков контроллера, а также сигналов на шинах данных и управления.

Одновременно с этим определенные клавиши клавиатуры ЭВМ станут органами для изменения информации на шине данных (например, клавиши «0», «1», ..., «7») и линии «Вывод» (например, клавиша «В»). Установив клавишами «0», ..., «7» содержимое выводимого байта данных (содержимое буферного регистра *A1*), можно (нажатием кнопки «В») дать контроллеру сигнал на преобразование этого байта в последовательность битов и передачу их (вместе со стартовым и стоповым битами) в ВУ. Эти действия (они подробно описаны в параграфе 8.4) легко прослеживаются по изменению (на экране телевизора) содержимого регистров и счетчиков моделируемого контроллера.

Легко заметить, что функциональные модели чрезвычайно универсальны: одни и те же технические средства позволяют производить изучение процесса преобразования информации в любых цифровых устройствах и с любой степенью детализации. Для исследования нового устройства надо только сменить моделирующую программу, программу вывода на экран

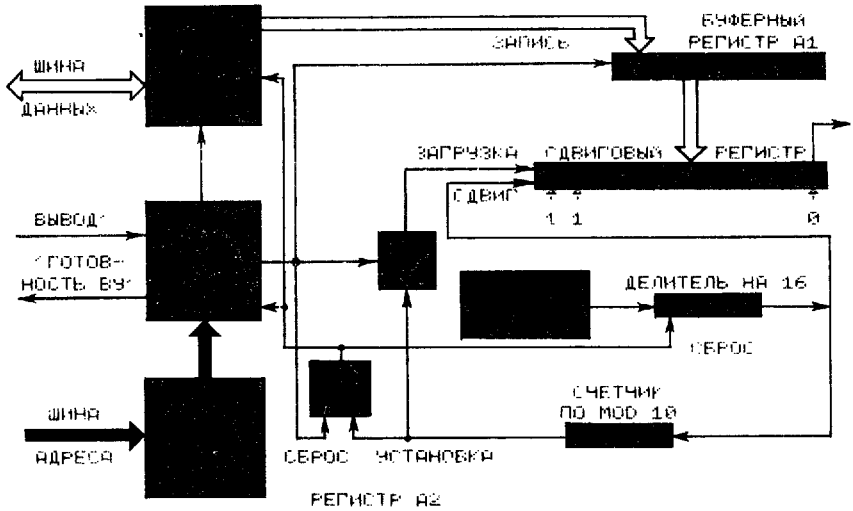


Рис. 9.1. Контроллер асинхронного последовательного вывода

изображения структуры моделируемого устройства и программу отображения состояния компонентов этого устройства. Основной недостаток функциональных моделей — невозможность исследования работы оригинала при реальных входных сигналах, нагрузках и помехах — присущ и физическим моделям, так как последние строятся на других (по сравнению с оригиналом) элементах, в другом конструктивном исполнении и к ним подключаются лишние устройства (схемы индикации состояний компонентов оригинала).

В соответствии со своим назначением функциональная модель должна правильно и полно отражать состояния лишь индицируемых компонентов оригинала и только в индицируемые моменты времени. Поэтому функциональное моделирование простейших цифровых устройств можно выполнять с помощью программы, которая по сигналам органов управления (например, клавиатуры персональной ЭВМ) выбирает из памяти и «распечатывает» на экране телевизора соответствующие им состояния компонентов оригинала.

Следовательно, при создании функциональной модели надо в первую очередь заботиться о способах ввода управляющих сигналов и вывода информации о состояниях моделируемого устройства, т. е. о факторах, определяющих необходимые уровень и полноту исследования. Та же часть программного обеспечения модели, с помощью которой анализируются текущее ее состояние, наличие сигналов, инициирующих переход в следующее состояние, и определяется новое содержимое индицируемых компонентов оригинала, может быть построена любым удобным (или доступным) для разработчика способом.

9.2. Взаимодействие с функциональной моделью

Если функциональная модель цифрового устройства реализуется с помощью персональной ЭВМ, то взаимодействие с такой моделью проще и дешевле всего организовать через клавиатуру этой ЭВМ и ее телевизионный экран (дисплей). Аналогичным образом может быть организовано взаимодействие с моделями, построенными на многотерминальной вычисли-

тельной системе (в дисплейном классе). Однако такой путь обладает рядом недостатков.

1. Телевизионный экран накладывает серьезные ограничения на объем выводимой информации. Например, при изучении принципа асинхронного последовательного обмена информацией на экране необходимо показать структуру не только передатчика (см. рис. 9.1), но и приемника информации (см. рис. 8.11), что трудно сделать, так как на экране можно изобразить лишь 24 строки по 64 или 80 символов в каждой. Кроме того, многие из находящихся в эксплуатации ЭВМ могут выводить на экран лишь буквы, цифры и символы в черно-белом изображении, что резко снижает наглядность создаваемых моделей.

2. Использование штатной клавиатуры персональной ЭВМ (клавиатуры, поставляемой с дисплеем) затрудняет общение с моделью, так как приходится запоминать, на какую функцию запрограммирована та или иная клавиша. Несколько лучше обстоит дело в тех случаях, когда клавиатура имеет ряд функциональных клавиш, программируемых на выполнение нужных действий.

Следовательно, только специально изготовленный пульт управления моделью и индикации ее состояний позволит организовать наиболее эффективное взаимодействие с моделью и получить всю необходимую информацию о ее функционировании.

На рис. 9.2, *а* приведен внешний вид, а на рис. 9.2, *б* — структурная схема пульта управления и индикации, используемого для изучения работы базовой ЭВМ. Пульт состоит из набора тумблеров и кнопок (см. рис. 3.1), индикатора матричного газоразрядного (ИМГ-3), контроллера для связи клавиатуры и индикатора с управляющей (моделирующей) ЭВМ.

ИМГ-3 — газоразрядная панель с устройством управления, которое может зажечь любой из 4096 (64×64) элементов изображения (ЭИ) размером 3×3 мм, расположенных на расстоянии 3 мм друг от друга. На панель накладывается маска со структурой базовой ЭВМ (см. рис. 3.1) или ее микропрограммным устройством управления (см. рис. 4.6). В изображениях

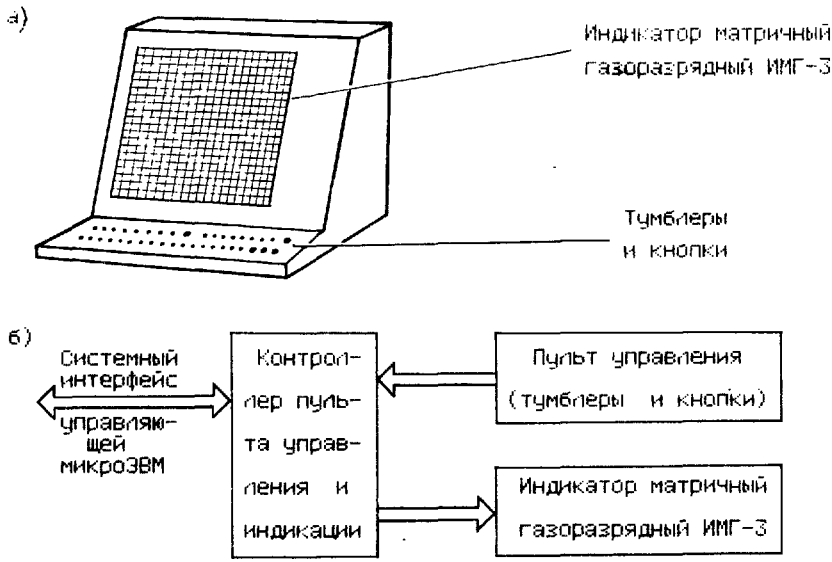


Рис. 9.2. Пульт управления и индикации функциональных моделей микроЭВМ:

а — внешний вид; *б* — структурная схема

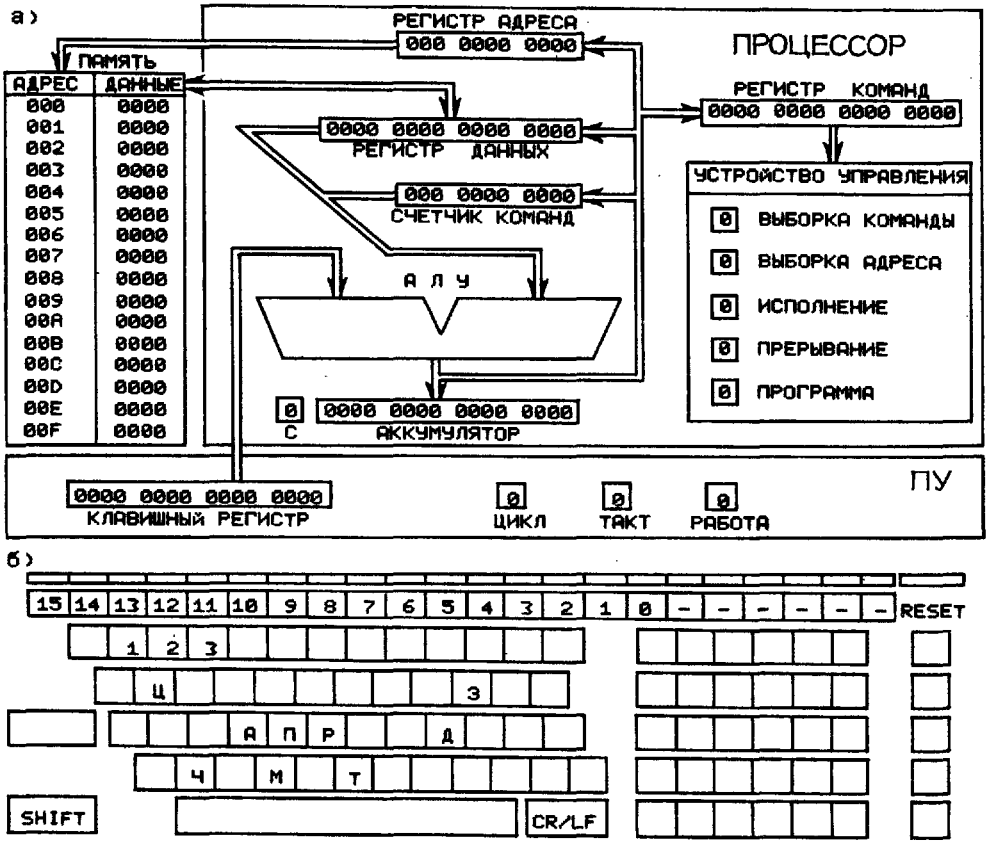


Рис. 9.3. Средства ПЭВМ «Искра 226», используемые для работы с моделью базовой ЭВМ:
а — экран со структурой процессора; *б* — клавиатура

регистров и других устройств этих масок пробиты круглые отверстия, расположенные над ЭИ панели. Моделирующая программа зажигает ярким зеленым цветом ряд ЭИ, соответствующих единичным состояниям тех или иных разрядов устройств базовой ЭВМ.

Такой пульт можно с успехом использовать для изучения процессов функционирования любых микроЭВМ. Он позволяет индицировать содержимое отдельных устройств для достаточно сложных структур. Органы управления пульта схожи с органами управления мини-ЭВМ, что создает при работе иллюзию общения с аппаратно реализованной ЭВМ, а не с ее программной моделью. Однако изготовление подобного пульта и устройств для его связи с моделирующей ЭВМ — достаточно трудоемкая задача. Поэтому далее будут рассмотрены некоторые приемы, позволяющие использовать для управления работой модели и индикации ее состояний штатные ВУ моделирующей ЭВМ — клавиатуру и дисплей.

Переключатели и кнопки. В составе штатных клавиатур моделирующих персональных ЭВМ (клавиатур, поставляемых с дисплеями) практически нет многопозиционных переключателей и очень мало двухпозиционных переключателей. Поэтому для имитации подобных переключателей моделируемых цифровых систем приходится использовать кнопки (клавиши, не имеющие механической фиксации положения) и часть экрана дисплея моделирующей ЭВМ.

На экране вычерчивается схема переключателя или высвечиваются символы, характеризующие текущее его состояние, а каждое нажатие соответствующей клавиши (кнопки) инициирует изменение состояния переключателя и его изображения на экране. Изменение состояния моделируемого переключателя осуществляется программным путем: получив сигнал от нажатой клавиши, программа анализирует текущее ее «положение» и изменяет его либо на противоположное (двухпозиционный переключатель), либо, например, на следующее по порядку (многопозиционный переключатель).

Так, для задания адреса ячейки памяти или ввода данных в модель базовой ЭВМ, построенной на персональной ЭВМ «Искра 226», используются функциональные клавиши клавиатуры этой персональной ЭВМ (рис. 9.3, *б*). С помощью каждой из клавиш можно изменить состояние одного разряда клавишного регистра, изображенного на экране дисплея (рис. 9.3, *а*). Если какая-то клавиша находилась в «положении 0» (в соответствующем разряде клавишного регистра высвечивается символ «0»), то после ее однократного нажатия клавиша будет переведена в «положение 1» (символ «0» заменится на символ «1»). Следующее нажатие той же клавиши переведет ее в «положение 0», т. е. вновь инвертирует соответствующий разряд клавишного регистра. Фрагменты моделирующей программы, используемые для осуществления описанных операций, имеют следующий вид:

```
10 DIM R¤1,K¤2,K¤(16)1,...
15 REM ОПИСАНИЕ РАБОЧЕЙ ПЕРЕМЕННОЙ R¤, А ТАКЖЕ ДВОИЧНОЙ ПЕРЕМЕННОЙ K¤ И
       СИМВОЛЬНОГО МАССИВА K¤(),  ИСПОЛЬЗУЕМОГО ДЛЯ СОХРАНЕНИЯ ТЕКУЩЕГО
       СОДЕРЖИМОГО КЛАВИШНОГО РЕГИСТРА ("АДРЕС/ДАННЫЕ")
200 KEYIN R¤,240,210:GOTO 300
205 REM ПРОВЕРКА СОСТОЯНИЯ КЛАВИАТУРЫ
210 IF R¤>HEX(0F)THEN 220:GOSUB´ 50:GOTO 300
215 REM,   ОБРАЩЕНИЕ К ПОДПРОГРАММЕ "КЛАВИШНЫЙ РЕГИСТР",
        ЕСЛИ НАЖАТА ОДНА ИЗ 16 СПЕЦКЛАВИШ С КОДАМИ 00 - FF
         ...

1000 DEFFN´ 50
1005 REM  ПОДПРОГРАММА "КЛАВИШНЫЙ РЕГИСТР"
1010 I=VAL(R¤)+1
1015 REM   ОПРЕДЕЛЕНИЕ НОМЕРА НАЖАТОЙ КЛАВИШИ
1020 IF K¤(I)="0"THEN1030:K¤(I)="0":GOTO 1040
1030 K¤(I)="1"
1035 REM ИНВЕРТИРОВАНИЕ I-ГО ЭЛЕМЕНТА МАССИВА K¤()
1040 FOR J=1TO2:R=0:FOR I=1TO8:IF K¤(I)="0"THEN 1050:R=R+2^(8-I)
1050 NEXT I:BIN(STR(K¤,J,1))=R:NEXT J
1055 REM  ОПРЕДЕЛЕНИЕ НОВОГО ЗНАЧЕНИЯ ДВОИЧНОЙ ПЕРЕМЕННОЙ K¤
1060 PRINT AT (22,8):FOR J=1TO4:FOR I=1TO4:PRINT K¤((J-1)*4+1);
     NEXT I:PRINT " ";:NEXT J
1065 REM ПОТЕТРАДНАЯ ПЕЧАТЬ СОДЕРЖИМОГО КЛАВИШНОГО РЕГИСТРА
1070 RETURN
```

Индикация. Эффективность изучения работы цифровой вычислительно-(управляющей) системы в значительной мере зависит от формы представления информации о состоянии ее устройств в интересующие исследователя моменты времени. Поэтому построение модели следует начинать с созда-

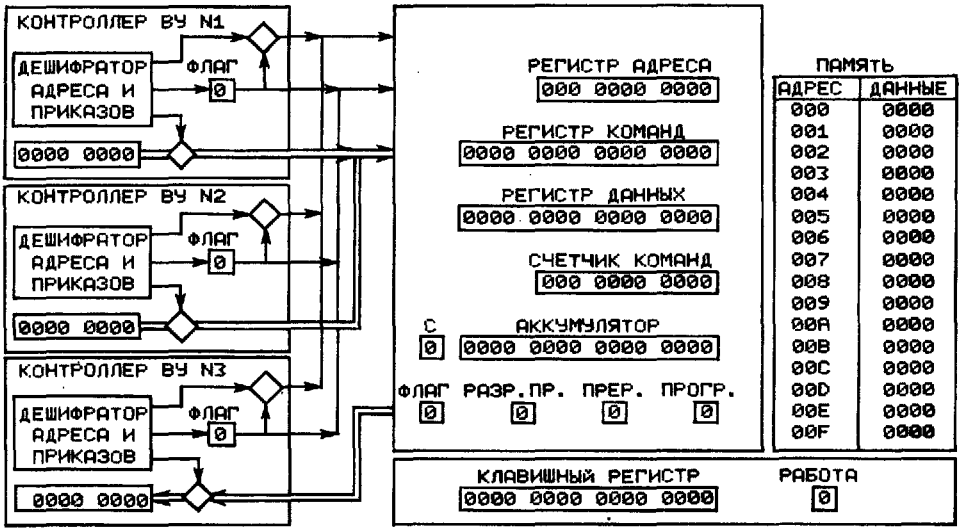


Рис. 9.4. Экран ПЭВМ «Искра 226» со структурой, используемой при изучении обмена данными между процессором и внешними устройствами базовой ЭВМ

ния достаточно подробной структурной схемы исследуемой системы и перечня тех ее устройств и связей между ними, состояние которых должно индицироваться на каждом из отображаемых шагов работы системы. Если изображение всех этих устройств и связей не может быть размещено на экране дисплея, то необходимо либо несколько упростить отображаемую структуру, либо перейти к ее поэтапному исследованию. Последнее даже предпочтительнее, так как чересчур подробная структура сложна для восприятия.

Например, экран персональной ЭВМ «Искра 226» не позволяет разместить изображения базовой ЭВМ, показанного на рис. 3.1. Поэтому исследо-

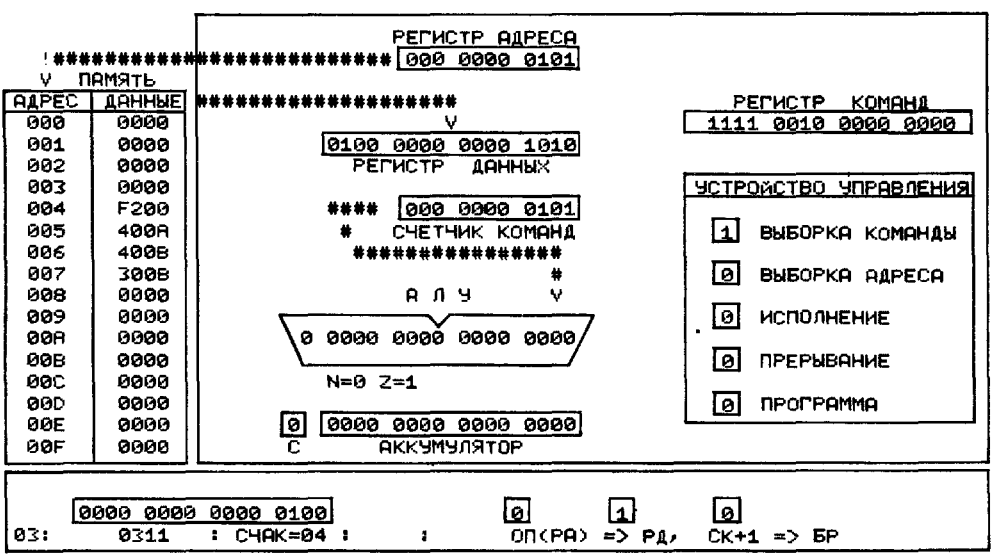


Рис. 9.5. Экран ПЭВМ «Искра 226» со структурой, используемой при изучении микропрограммного управления базовой ЭВМ

вание способов обмена данными между процессором и внешними устройствами базовой ЭВМ выполняется по структуре, показанной на рис. 9.4, отличающейся от структуры на рис. 9.3, используемой для исследования работы процессора базовой ЭВМ.

При отсутствии символьно-графического дисплея структура исследуемой цифровой системы может быть изображена на маске, накладываемой на экран символьного дисплея. Такую маску лучше всего нарисовать на прозрачной пленке, но можно и на бумаге, в которой затем прорезаются отверстия для индицирования содержимого устройств исследуемой системы.

Это содержимое целесообразнее всего выводить в виде состояния отдельных разрядов индицируемых устройств. Такое состояние можно показать, например, символами «*» и « » (пробел) или «0» и «1». Однако для уменьшения площади изображаемой структуры возможен и вывод в форме шестнадцатеричных чисел (так выводится содержимое ячеек памяти базовой ЭВМ на рис. 9.3 и 9.4).

Потактовую работу системы нагляднее всего иллюстрировать с помощью индикации тех связей, которые устанавливаются между отдельными устройствами системы в каждом из ее состояний. Подобным образом организуется изучение микропрограммного управления базовой ЭВМ (рис. 9.5).

9.3. Построение функциональной модели

Назначение функциональной модели (см. параграф 9.1) предопределяет основные режимы ее работы: пошаговый режим и режим достаточно медленного выполнения небольших программ, в процессе которого можно проследить характер изменения состояний всех индицируемых устройств моделируемой вычислительной системы. Следовательно, критерий «минимальное время работы» не является основным при разработке моделирующих программ. Такие программы можно писать на языке высокого уровня даже тогда, когда они будут при работе интерпретироваться на моделирующей ЭВМ.

Создание функциональной модели целесообразно начинать с программ, обеспечивающих взаимодействие с исследователем: программ обработки управляющих сигналов и ввода данных с клавиатуры моделирующей ЭВМ, а также программ вычерчивания изучаемой структуры и вывода состояний ее устройств.

Наименее жесткие требования предъявляются к программам вычерчивания исследуемой структуры, так как эта операция осуществляется перед началом исследования и может занимать достаточно большой промежуток времени (десятки секунд). При создании таких программ для персональных ЭВМ, как правило, не возникает и серьезных ограничений на объем используемой ими памяти (ее незачем экономить при однопрограммном режиме работы ЭВМ).

Программы обработки управляющих сигналов и ввода данных (команд) с клавиатуры моделирующей ЭВМ надо проектировать и размещать среди остальных программ моделирующего комплекса так, чтобы промежуток времени между нажатием какой-либо клавиши и соответствующим изменением рисунка структуры изучаемой системы не превышал долей секунды. В тех же случаях, когда модели требуется больший промежуток времени для обработки приказа, поданного с клавиатуры, целесообразно как-то информировать исследователя о том, что приказ получен и модель приступила к его реализации. Это можно делать путем высвечивания трафарета.

Например, в модели базовой ЭВМ после нажатия клавиш, имитирующих работу кнопок «ВВОД АДРЕСА», «ЗАПИСЬ», «ЧТЕНИЕ», «ПУСК» и «ПРОДОЛЖЕНИЕ», рядом с изображением клавишного регистра выво-

дятся на белом фоне соответствующие надписи: «АДР.», «ЗАП.», «ЧТЕН.», «ПУСК» и «ЖДИТЕ». Трафарет перестает высвечиваться после завершения всех действий по обработке приказа и вывода на экран нового состояния модели.

Вывод состояний устройства исследуемой системы можно осуществлять:

по мере изменения этих состояний;

после завершения всех действий по реализации заданного шага (например, микрокоманды или команды);

в смешанном режиме.

Первый способ целесообразен лишь для простых моделей, в которых на каждом из заданных шагов производится однократное изменение состояния каких-либо устройств. В противном случае время выполнения шага (или последовательности шагов) существенно возрастет, так как печать информации (даже на экране дисплея) — наиболее длительная из моделирующих операций.

Однако не всегда выгодно запоминать, какие устройства изменяли свое состояние на том или ином шаге, чтобы в конце шага вывести на экран их новое содержимое. Иногда целесообразнее в конце шага выводить на экран содержимое всех устройств модели независимо от того, изменяли ли они на этом шаге свое состояние или не изменяли. Следовательно, выбор того или иного режима вывода информации определяется характером построения модели. В модели базовой ЭВМ используется смешенный режим: содержимое индицируемых регистров процессора и ячеек памяти выводится в конце заданного шага (микрокоманды, машинного цикла или команды), а остальных устройств — по мере изменения их состояний.

Содержание программ, обеспечивающих переработку информации в функциональной модели, определяется не только сложностью оригинала, но и тем, какие устройства оригинала и с каким шагом дискретизации должны индицироваться при работе модели. Так, программы для моделирования микроЭВМ с возможностью исследования ее работы в покомандном режиме значительно проще, чем аналогичные программы, в которых дополнительно предусматривается исследование реализации отдельных команд микроЭВМ по машинным циклам и тактам. Это связано с тем, что функциональная модель должна правильно воспроизводить состояния оригинала лишь в момент вывода на экран содержимого индицируемых устройств. Поэтому в первом случае совсем не обязательно воспроизводить микропрограммный уровень работы оригинала, что позволяет упростить моделирующие программы.

9.4. Исследования, проводимые на функциональной модели

Как было сказано выше, на функциональных моделях нельзя исследовать лишь те аспекты работы оригинала, которые связаны с преобразованием физических сигналов в реальном масштабе времени. Но наиболее удобным их применением является исследование принципов построения и функционирования вычислительных устройств.

Так, модель базовой ЭВМ используется для проведения довольно подробных исследований взаимодействия ее устройств при выборке и исполнении команд [23]. Для этого сначала предлагаются небольшие последовательности кодов, содержащих команды линейной (табл. 9.1) и разветвляющейся (табл. 9.2) программ, а также коды данных. Последовательность кодов заносится в память базовой ЭВМ (см. параграф 3.1), в счетчик команд заносится пусковой адрес (адрес команды, отмеченной символом «+»), машина устанавливается в режим покомандного исполнения про-

Таблица 9.1

Линейная программа

Ячейки памяти		Ячейки памяти	
Адрес	Содержимое	Адрес	Содержимое
017	0000	01E	301A
018	0255	01F	F200
019	7C99	020	4019
01A	0000	021	101A
01B	+F200	022	301A
01C	4024	023	F000
01D	6018	024	C3CF

Таблица 9.2

Программа с ветвлением

Ячейки памяти		Ячейки памяти	
Адрес	Содержимое	Адрес	Содержимое
017	54F7	01E	F100
018	+F200	01F	F000
019	4022	020	3017
01A	4023	021	C01F
01B	9020	022	FFD1
01C	F200	023	002F
01D	3017	024	7C99

граммы, и производится ее пошаговое выполнение. Содержимое всех регистров процессора после исполнения каждой команды заносится в таблицу, аналогичную табл. 3.2, анализируется и комментируется.

Затем исследователь должен отразить: назначение программы; реализуемые ею функции (формулы); область представления исходных данных и результатов; расположение в памяти программы, исходных данных и результатов; адреса первой и последней команд программы и т. п. Сходным образом (по готовой или составленной программе) изучается эффективность различных способов организации циклических программ (тех средств базовой ЭВМ, которые позволяют строить такие алгоритмы), работы с подпрограммами (способы передачи параметров) и т. п. Если при этом используется модель, построенная на персональной ЭВМ «Искра 226», то на экране последней вычерчивается структура, показанная на рис. 9.3, *а*.

С помощью структуры, представленной на рис. 9.4, исследуются алгоритмы обмена информацией с периферийным оборудованием (асинхронного обмена и обмена по прерыванию программы). Исследователь играет роль устройств ввода информации, т. е. устанавливает с помощью клавиатуры коды вводимых символов и дает сигнал о готовности (устанавливает флаг контроллера).

Наконец, с помощью структуры на рис. 9.5 производится изучение функционирования базовой ЭВМ на микропрограммном уровне. Для этого предлагается сначала ознакомиться с потактовым выполнением нескольких команд машины, а затем написать, ввести в память микрокоманд и испол-

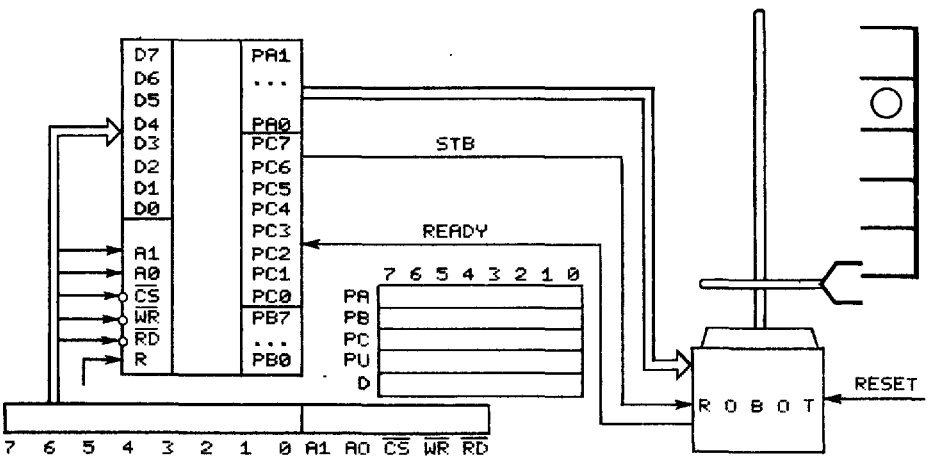


Рис. 9.6. Экран ПЭВМ «Искра 226» со структурой, используемой при изучении микросхемы параллельного интерфейса КР580ВВ55

нить несколько новых команд (команд с кодами операций 7, D, FC, FD, FE и FF). Такими командами могут быть команды: загрузки аккумулятора (код операции 7), перехода по четности или нечетности (код операции D), получения дополнительного кода содержимого аккумулятора (код операции FC) и т. п.

Функциональные модели очень удобны для иллюстрации последовательного обмена данными между ЭВМ и ВУ (см. параграфы 7.1 и 8.4). В них легко показать воздействия помех на искажение передаваемых данных, работу схем контроля по четности (нечетности), ошибки из-за рассинхронизации генераторов при организации асинхронного последовательного обмена и т. п.

Хорошо иллюстрируется и работа серийных контроллеров. Так, для исследования работы параллельного интерфейса КР580ВВ55 используется модель со структурой, представленной на рис. 9.6. С одной стороны, интерфейс КР580ВВ55 связан с процессором, роль которого играет исследователь, а с другой — с моделью складского робота. Через него передаются приказы на выполнение какого-либо действия робота (вверх, вниз, вперед, назад, сжать, расжать). Внутренняя структура интерфейса КР580ВВ55 представлена в виде набора выходных регистров портов ввода-вывода, регистра управления и буферного регистра данных.

Модель параллельного интерфейса программируется исследователем в нужный для обмена режим. Затем исследователь записывает в параллельный интерфейс приказ, который должен быть принят роботом, и два слова, формирующие строб. Приняв строб, робот считывает приказ и выполняет одно из вышеперечисленных действий. Для перемещения детали из одной ячейки в другую исследователю (моделирующему работу процессора) необходимо передать от 20 до 30 8-разрядных слов. Сбои, возникающие из-за неверной команды или других причин, устраняются путем передачи сигнала «Сброс управления робота».

Чрезвычайно просто создавать модели отдельных устройств процессора. На них очень наглядно показывается работа АЛУ, сдвигателей, различных комбинационных схем и т. п.

Часть II

СОВРЕМЕННЫЕ СРЕДСТВА МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ И ПЕРСПЕКТИВЫ ИХ РАЗВИТИЯ

Глава 10

МИКРОПРОЦЕССОРНЫЕ КОМПЛЕКТЫ ИНТЕГРАЛЬНЫХ СХЕМ

10.1. Классификация микропроцессорных комплектов интегральных схем

Все современные микроЭВМ, а также многие устройства, блоки и узлы ЭВМ других классов построены с использованием больших интегральных схем, входящих в различные микропроцессорные комплекты (МПК). В настоящее время под МПК БИС понимается совокупность микропроцессорных и других интегральных микросхем, совместимых по технологическому и конструктивному исполнению и предназначенных для совместного использования при построении различных средств вычислительной техники: микропроцессоров, микроконтроллеров, микроЭВМ и т. п. [24].

Основой каждого МПК БИС является микропроцессор — программно-управляемое устройство для обработки данных, построенное на одной или нескольких БИС. Однако одного только микропроцессора недостаточно для построения как микроЭВМ, так и различных устройств управления на их основе. Любая микропроцессорная система должна включать средства для хранения программ, данных и результатов обработки данных, средства ввода-вывода информации, в том числе устройства для связи с объектами управления и средствами отображения результатов обработки. Все это обусловило необходимость разработки комплектов БИС, обеспечивающих потребности практики.

В настоящее время отечественная промышленность выпускает более десяти МПК БИС, различающихся набором БИС и их функциональной полнотой, конструкцией и технологией изготовления. Различия в МПК БИС обусловливают области их наиболее эффективного применения. Именно область применения МПК БИС может служить первым признаком их классификации. По этому признаку все МПК БИС можно разделить на две группы: универсальные МПК БИС и специализированные.

К универсальным МПК БИС относятся такие комплекты, на основе которых могут быть построены разнообразные средства вычислительной техники — от микроконтроллера до универсальных микроЭВМ различных типов. Эта группа МПК БИС наиболее многочисленна, и в нее входят такие комплекты, как КР580, К582, К587, К1804, К1810, и др.

Специализированные МПК БИС предназначаются для построения только одного типа микроЭВМ. В эту группу входят МПК БИС К536, К581, К1801 и др.

Следующим признаком классификации МПК БИС может служить количество микропроцессорных БИС в комплекте. Здесь можно выделить микропроцессоры одно-, многокристальные и с разрядно-модульной организацией (секционные микропроцессоры).

Однокристальный микропроцессор представляет собой функционально и конструктивно законченное устройство для обработки данных фиксированной разрядности, реализующее фиксированную систему команд. Примерами однокристальных микропроцессоров могут служить БИС КР580ИК80А и КМ1810ВМ86.

В многокристальном микропроцессоре отдельные БИС предназначены для выполнения строго определенного набора функций и могут работать автономно, а совместное их использование обеспечивает построение микропроцессора. Такой подход к построению микропроцессора позволяет создать более сложные в функциональном отношении отдельные узлы микропроцессора по сравнению с однокристальной реализацией микропроцессора. Многокристальные микропроцессоры имеют фиксированную разрядность. Система команд таких микропроцессоров может быть как фиксированной, так и изменяемой в случае, если память микропрограмм реализуется в отдельной БИС. Типичным примером многокристального микропроцессора является МПК БИС серии К581.

Микропроцессоры с разрядно-модульной организацией состоят из БИС, которые предназначены для обработки нескольких разрядов данных или выполнения определенных управляющих операций. Секционность БИС обеспечивает возможность наращивания разрядности обрабатываемых данных или усложнения устройств управления микропроцессором при параллельном включении БИС. Параллельное включение микропроцессорных секций реализуется специальными БИС, также входящими в микропроцессорный комплект. Примерами отечественных микропроцессоров с разрядно-модульной организацией могут служить БИС К589ИК02 и К1804ВС1.

Приведенная здесь классификация представляется вполне достаточной для первого знакомства с МПК БИС. Более подробно эти вопросы освещены в работах [12, 24]. Ниже рассмотрены наиболее распространенные в отечественной практике и относящиеся к различным классам микропроцессорные комплекты БИС.

10.2. Микропроцессорный комплект БИС КР580

Универсальный микропроцессорный комплект БИС серии КР580 является в настоящее время наиболее распространенным. БИС этого комплекта применяют для построения 8-разрядных микроЭВМ и микроконтроллеров. Широкое использование БИС серии КР580 обусловлено, в первую очередь, функциональной законченностью комплекта БИС в целом. В состав комплекта вместе с БИС 8-разрядного микропроцессора входит еще целый ряд БИС, обеспечивающих все необходимые в микроЭВМ способы и режимы обмена с ВУ, управление некоторыми ВУ и формирование части управляющих сигналов системного интерфейса микроЭВМ:

1) микропроцессор КР580ИК80А;
2) программируемый параллельный интерфейс КР580ВВ55;
3) программируемый последовательный интерфейс КР580ВВ51;
4) программируемый контроллер прерываний КР580ВН59;
5) программируемый контроллер прямого доступа к памяти КР580ВТ57;
6) программируемый таймер КР580ВИ53;
7) программируемый контроллер электронно-лучевой трубки КР580ВГ75;
8) программируемый контроллер клавиатуры и индикации КР580ВВ79;

9) системные контроллеры и шинные формирователи КР580ВК28 и КР580ВК38;

10) буферные регистры КР580ИР82 и КР580ИР83;

11) шинный формирователь КР580ВА86;

12) генератор тактовых импульсов КР580ГФ24.

Рассмотрим кратко назначение некоторых БИС серии КР580. Более подробное описание микропроцессорного комплекта БИС КР580, способов использования отдельных БИС приведено в целом ряде книг советских и зарубежных авторов [1, 3, 12, 25].

Микропроцессор КР580ИК80А. Этот конструктивно и функционально законченный модуль предназначен для построения микро-ЭВМ с изолированными шинами адреса, данных и управления (см. параграф 5.2). На основе БИС КР580ИК80А построены микроЭВМ СМ1800, «Электроника К1-20», интерфейсные функциональные блоки ЭВМ «Искра 226».

Данный микропроцессор (МП) имеет фиксированный набор команд, перечень которых совместно с используемыми режимами адресации приведен в приложении Г. Перечень шин и функциональное назначение управляющих выводов микропроцессора представлен на рис. 8.3.

Ограниченное число выводов БИС МП, отведенных под управляющие сигналы, привело к тому, что в самом микропроцессоре формируется только часть управляющих сигналов, необходимых для функционирования микроЭВМ. Формирование системных управляющих сигналов для обращения к памяти и ВУ производится во внешних по отношению к МП схемах, например: в системных контроллерах КР580ВК28 и КР580ВК38 или специальных регистрах. Для этого в регистр загружается слово состояния МП — управляющее слово. Назначение отдельных разрядов слова состояния приведено в табл. 8.1. Слово состояния загружается во внешний регистр в начале каждого машинного цикла и определяет его тип — один из 10 машинных циклов, которые могут быть инициированы при выполнении команды в данном микропроцессоре.

Программируемый параллельный интерфейс КР580ВВ55. Обмен данными с ВУ в параллельном формате исключительно широко используется в микроЭВМ (см. параграф 8.3). Поэтому создание БИС, управляющей параллельным обменом данными, явилось естественным шагом в развитии средств микропроцессорной техники.

БИС КР580ВВ55 обеспечивает подключение ВУ к шине данных системного интерфейса микроЭВМ через три независимых двунаправленных канала (порта) ввода-вывода: *A*, *B* и *C* (рис. 10.1, *б*). Связь БИС с системным интерфейсом микроЭВМ осуществляется по шине данных D_7, ..., D_0.

Для адресации внутренних регистров БИС используются линии A_1 и A_0 шины адреса системного интерфейса: 00 — порт *A*, 01 — порт *B*, 10 — порт *C* и 11 — регистр управления.

Управление обменом данными БИС с системным интерфейсом производится с помощью сигналов *CS*, *RD* и $\overline{WR}$. Управляющий сигнал *RESET* устанавливает БИС в начальное состояние, в котором регистр управления обнуляется, а все три порта переводятся в режим ввода.

Программирование БИС (настройка на требуемый режим обмена) и непосредственно обмен данными производятся с помощью команд IN (ввод) и OUT (вывод) микропроцессора. При программировании БИС в регистр управления командой OUT записывается управляющее слово (приказ), которое определяет один из трех режимов работы параллельного интерфейса.

Режим 0 — программно-управляемый ввод-вывод данных в синхронном режиме по трем 8-разрядным каналам *A*, *B* и *C*. Каждый канал может

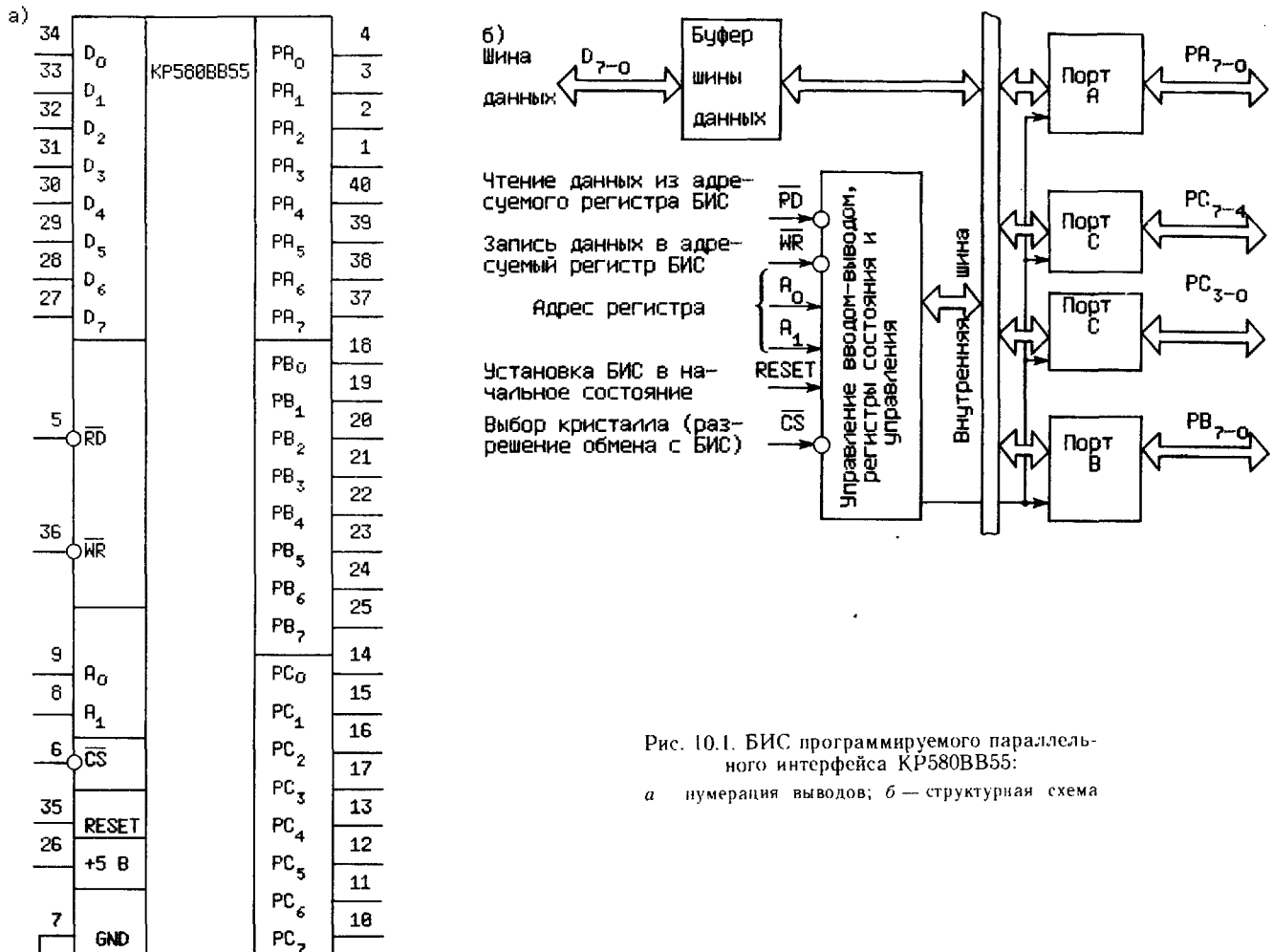


Рис. 10.1. БИС программируемого параллельного интерфейса КР580ВВ55:

а — нумерация выводов; *б* — структурная схема

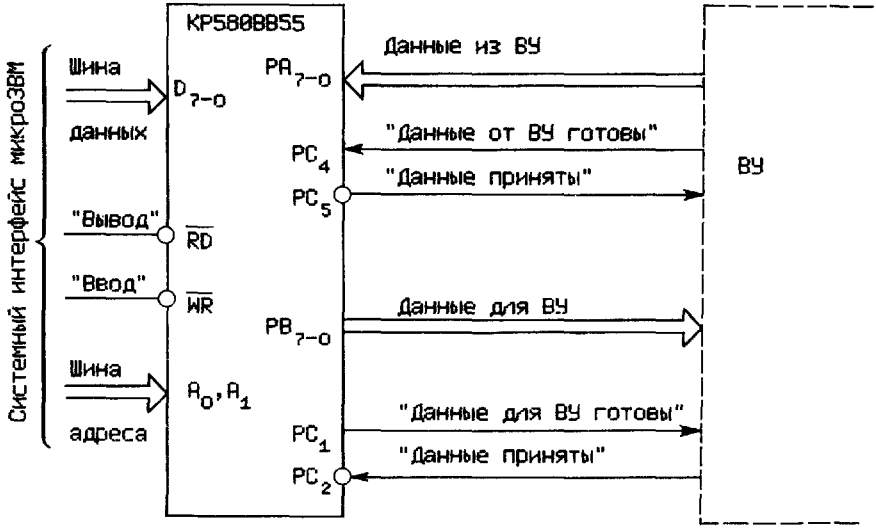


Рис. 10.2. Параллельный интерфейс (контроллер) ввода-вывода на БИС КР580ВВ55

быть использован только для ввода или вывода информации, а канал *C* дополнительно может быть разделен на два 4-разрядных канала.

Режим 1 — обмен данными с ВУ по каналам *A* и *B* в асинхронном режиме и режиме прерывания программы. Управляющие сигналы при этом передаются по каналу *C*.

Режим 2 — двунаправленный обмен данными с ВУ по каналу *A* в режиме прерывания программы. Канал *C* используется для передачи управляющих сигналов.

Пример 10.1. Один из возможных вариантов реализации контроллеров параллельного ввода и вывода (см. рис. 8.6 и 8.7) посредством одной БИС КР580ВВ55 приведен на рис. 10.2.

В качестве регистров данных контроллеров ввода и вывода используются порты A и B КР580ВВ55. Регистрами состояния контроллеров служат разряды порта C: PC_2 — при выводе данных, PC_4 — при вводе данных. Формирование управляющих сигналов для шины связи с ВУ также осуществляется с использованием разрядов порта C: PC_1 — при выводе данных, PC_5 — при вводе данных.

Программируемый последовательный интерфейс КР580ВВ51. Эта БИС имеет еще одно название – универсальный синхронно-асинхронный приемопередатчик. В этом названии полностью отражено функциональное назначение БИС КР580ВВ51. Таким образом, посредством этой БИС можно реализовать все рассмотренные в параграфе 8.4 режимы последовательного обмена данными с ВУ.

БИС КР580ВВ51 (рис. 10.3) имеет в своем составе два независимых последовательных интерфейса: ввода (приемник) и вывода (передатчик). Связь приемника и передатчика БИС с системным интерфейсом микроЭВМ осуществляется через двунаправленный буфер данных по шине данных $D_7, \ldots, D_0$.

Управление обменом данными с системным интерфейсом производится в БИС с помощью управляющих сигналов CS, $\overline{RD}$, $\overline{WR}$ и C/D. Подачей различных комбинаций управляющих сигналов инициируется чтение из БИС байта данных или слова состояния либо запись в БИС байта данных, слова режима, слова приказа или символа синхронизации.

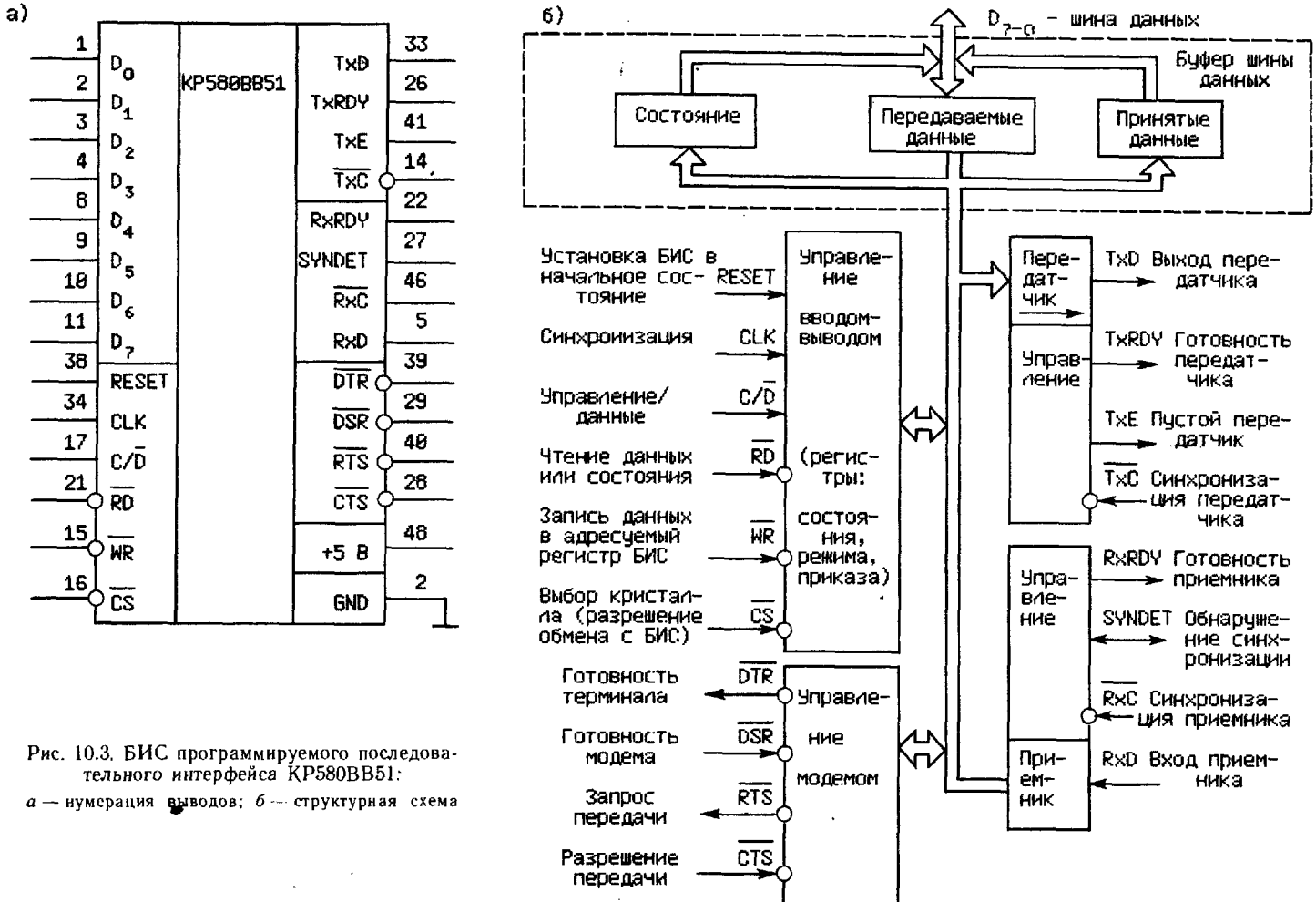


Рис. 10.3. БИС программируемого последовательного интерфейса КР580ВВ51:

а — нумерация выводов; *б* — структурная схема

Схема управления БИС служит для формирования внутренних управляющих сигналов из сигналов системного интерфейса. Она содержит регистры слова приказа и слова режима, содержимое которых определяет режим работы БИС: синхронный или асинхронный. В слове режима задается также скорость передачи, длина передаваемого символа, количество стоп-битов, наличие либо отсутствие проверки на четность (нечетность).

Программирование БИС (настройка на требуемый режим обмена) производится записью управляющих слов в указанные регистры БИС в определенном порядке. Слово режима записывается в БИС сразу после ее установки в исходное состояние сигналом RESET. Слово приказа подается сразу за словом режима (при программировании асинхронного обмена) либо после загрузки символа (символов) синхронизации (при программировании синхронного обмена).

Синхронизация работы БИС осуществляется сигналами Φ_2 генератора тактовой частоты микроЭВМ, подаваемыми на вход CLK.

С помощью управляющих сигналов $\overline{DTR}$, $\overline{DSR}$, $\overline{RTS}$ и $\overline{CTS}$ БИС может управлять работой модема. Выходные сигналы $\overline{DTR}$ и $\overline{RTS}$ задаются в слове приказа, а входные $\overline{DSR}$ и $\overline{CTS}$ — считываются в слове состояния.

Программируемый контроллер прерываний КР580ВН59 (рис. 10.4). Используется он для организации в микроЭВМ многоуровневой векторной системы прерываний. Одна БИС КР580ВН59 обеспечивает прием и обработку восьми сигналов прерывания. Возможно совместное использование до восьми БИС, т. е. увеличение числа сигналов прерывания до 64. Любой сигнал или группа сигналов прерывания могут быть программно запрещены (замаскированы).

Связь БИС с системным интерфейсом микроЭВМ осуществляется через двунаправленный буфер шины данных по линиям D_7, ..., D_0 с использованием управляющих сигналов $\overline{RD}$, $\overline{WR}$, A_0 и $\overline{CS}$ (рис. 10.4, *б*). Через буфер данных в процессор передаются содержимое регистра состояния БИС и команда CALL, вызывающая переключение процессора на выполнение подпрограммы обработки сигнала прерывания с наивысшим приоритетом. Процессор передает в БИС управляющие слова — приказы инициализации и рабочие приказы, которые настраивают БИС на один из четырех режимов работы.

Вложенные прерывания. Каждому входу запроса прерывания IR_7, ..., IR_0 присваивается фиксированный приоритет в порядке изменения номеров: IR_0 — наивысший приоритет, IR_7 — самый низкий приоритет. Запрос прерывания с большим приоритетом прерывает обслуживание прерываний с меньшим приоритетом.

Циклический приоритет. Каждому входу IR_7, ..., IR_0 присваивается приоритет, как и в предыдущем случае, но после обслуживания какого-либо запроса прерывания приоритеты всех входов изменяются в циклическом порядке так, что последнему обслуженному входу присваивается низший приоритет.

Адресуемые приоритеты. Этот режим аналогичен режиму циклических приоритетов с той разницей, что вход, которому присваивается низший приоритет, задается программно.

Режим опроса. В этом режиме прерывания процессора запрещены, а идентификация запросов прерывания производится после считывания в процессор состояния БИС.

Для всех режимов прерывания идентификация сигналов прерывания, поступающих в процессор через БИС КР580ВН59, производится с помощью вектора прерывания — адреса начала подпрограммы обработки соответствующего сигнала прерывания. Старшие разряды (A_{15}, ..., A_5) вектора пре-

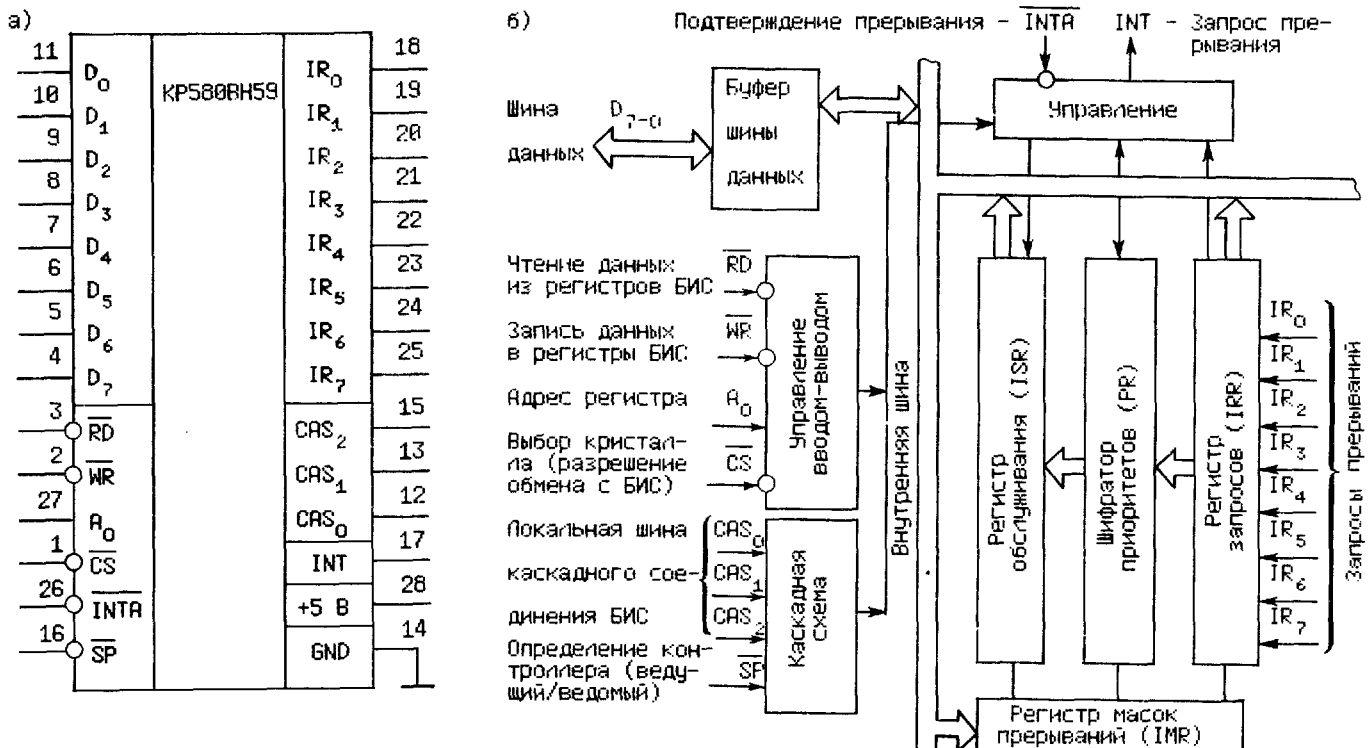


Рис. 10.4. БИС программируемого контроллера прерываний КР580ВН59:

а — нумерация выводов; *б* — структурная схема

рывания, общие для всех входов IR_7, ..., IR_0 контроллера, передаются в него приказом инициализации, а младшие разряды, индивидуальные для каждого входа, формируются непосредственно в контроллере. В ответ на сигнал запроса прерывания INT процессор выдает сигнал подтверждения прерывания $\overline{INTA}$. Сигнал $\overline{INTA}$ заставляет контроллер выдать на шину данных код операции команды CALL и затем в ответ на два дополнительных сигнала процессора $\overline{INTA}$ выдать два байта вектора прерывания. Процессор, получив от контроллера вектор прерывания, переключается на подпрограмму обработки прерывания.

Совместное использование нескольких БИС программируемого контроллера прерываний обеспечивается схемой каскадирования, с помощью которой можно объединить до восьми БИС. Выбор БИС, требующей обслуживания, осуществляется по линиям CAS_2, ..., CAS_0, объединяющим все контроллеры, а сигнал SP используется для определения контроллера как ведущего или ведомого.

Программируемый контроллер прямого доступа к памяти КР580ВТ57. Предназначен он для управления обменом данными между ВУ и основной памятью микроЭВМ в режиме прямого доступа к памяти (ПДП). Контроллер управляет процессом предоставления прямого доступа к памяти, формирует в процессе обмена последовательность адресов ячеек памяти и сигналы управления обменом.

В БИС КР580ВТ57 (рис. 10.5, *б*) реализованы четыре независимых канала ПДП K_0, ..., K_3. Управление работой каждого канала ПДП осуществляется с помощью двух 16-разрядных регистров: регистра начального адреса и регистра управления. В регистр начального адреса при программировании БИС заносится начальный адрес передаваемого массива данных. В 14 младших разрядах регистра управления размещается счетчик, в который заносится число на единицу меньше длины передаваемого массива данных, т. е. размер массива не может превышать 16 Кбайт. 15-й и 14-й разряды регистра управления определяют тип операции обмена: 00 — контроль; 01 — запись в память; 10 — чтение из памяти; 11 — запрещенное состояние.

Связь контроллера с системным интерфейсом микроЭВМ осуществляется по шине данных D_7, ..., D_0 через двунаправленный буфер шины данных. Запись информации в регистры начального адреса и управления каналов контроллера производится программным путем. Адресуются внутренние регистры контроллера по линиям A_3, ..., A_0.

Помимо указанных регистров в состав контроллера входят: регистр режима, определяющий общие функции контроллера, в том числе и приоритеты каналов ПДП;

регистр состояния, отображающий состояние окончания счета (сигнал TC), т. е. окончание передачи массива данных по соответствующим каналам ПДП.

Приоритеты входных запросов на предоставление прямого доступа устанавливаются программно записью информации в регистр приказов: либо фиксированные (DRQ_0 — высший, DRQ_3 — низший), либо циклические, т. е. последнему обслуженному запросу присваивается низший приоритет, а приоритеты остальных запросов изменяются в круговой последовательности.

Для формирования 16-разрядного адреса ячейки памяти на шине адреса системного интерфейса контроллеру прямого доступа требуется внешний 8-разрядный регистр. В этот регистр в начале цикла ПДП контроллер с использованием строба $ADSTB$ по шине данных D_7,..., D_0 записывает восемь старших разрядов адреса A_{15},...,A_8. Выходы внешнего регистра подключают-

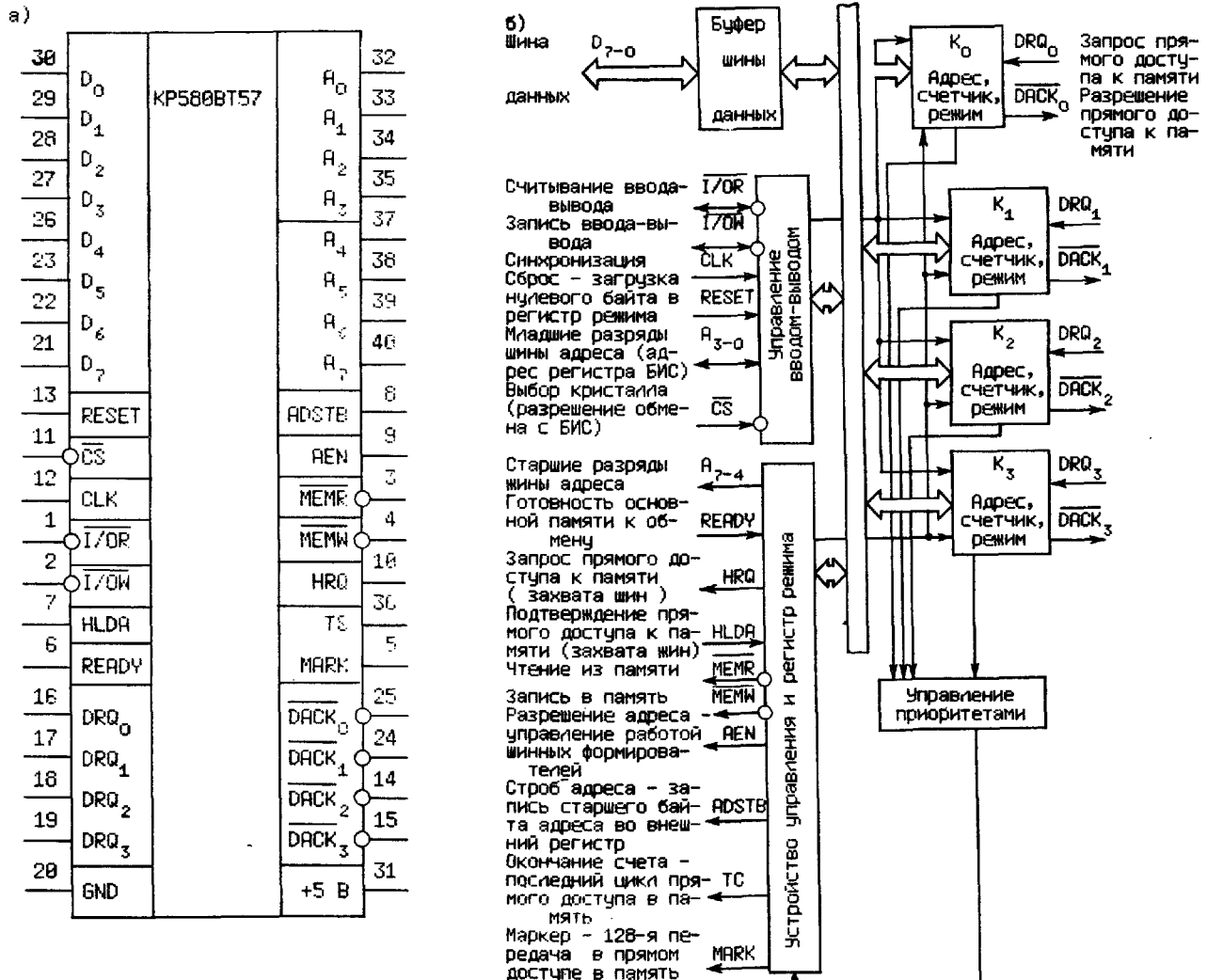

а)
КР580ВТ57
30 29 28 27 26 23 22 21 13 11 12 1 2 7 6 16 17 18 19 20
32 33 34 35 37 38 39 40 8 9 3 4 10 36 5 25 24 14 15 31
D0 D1 D2 D3 D4 D5 D6 D7
RESET
CS
CLK
I/OR
I/OW
HLDA
READY
DRQ0 DRQ1 DRQ2 DRQ3
GND
A0 A1 A2 A3 A4 A5 A6 A7
ADSTB
AEN
MEMR
MEMW
HRQ
TC
MARK
DACK0 DACK1 DACK2 DACK3
+5 В
б)
Шина данных
D7-0
Буфер шины данных
Считывание ввода-вывода I/OR
Запись ввода-вывода I/OW
Синхронизация CLK
Сброс - загрузка нулевого байта в регистр режима RESET
Младшие разряды шины адреса (адрес регистра БИС) A3-0
Выбор кристалла (разрешение обмена с БИС) CS
Управление вводом-выводом
Старшие разряды шины адреса A7-4
Готовность основной памяти к обмену READY
Запрос прямого доступа к памяти (захвата шин) HRQ
Подтверждение прямого доступа к памяти (захвата шин) HLDA
Чтение из памяти MEMR
Запись в память MEMW
Разрешение адреса - управление работой шинных формирователей AEN
Строб адреса - запись старшего байта адреса во внешний регистр ADSTB
Окончание счета - последний цикл прямого доступа в память TC
Маркер - 128-я передача в прямом доступе в память MARK
Устройство управления и регистр режима
K0 Адрес, счетчик, режим
DRQ0 Запрос прямого доступа к памяти
DACK0 Разрешение прямого доступа к памяти
K1 Адрес, счетчик, режим
DRQ1 DACK1
K2 Адрес, счетчик, режим
DRQ2 DACK2
K3 Адрес, счетчик, режим
DRQ3 DACK3
Управление приоритетами

Рис. 10.5. БИС программируемого контроллера прямого доступа к памяти КР580ВТ57:
а — нумерация выводов; *б* — структурная схема

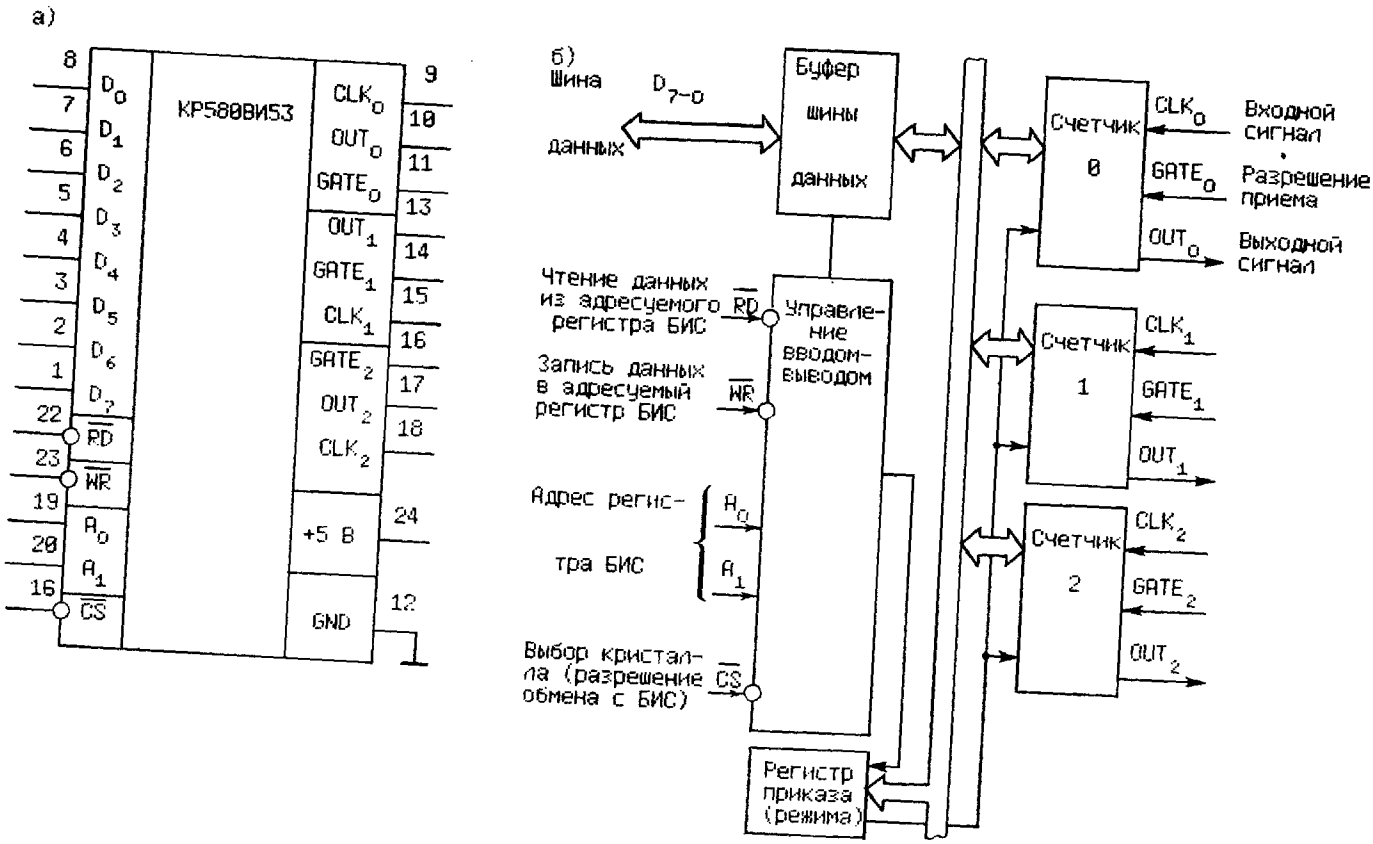


Рис. 10.6. БИС программируемого таймера КР580ВИ53:
а — нумерация выводов; *б* — структурная схема

ся к линиям $A_{15},...,A_8$ шины адреса системного интерфейса микроЭВМ, а младшие разряды адреса формируются на выходах A_7, ..., A_0 контроллера.

Программируемый таймер КР580ВИ53. Предназначен он для реализации таких довольно распространенных управляющих функций микроЭВМ, как формирование временных интервалов, подсчет числа внешних событий, генерация сигналов переменной частоты, и т. п.

В состав БИС КР580ВИ53 (рис. 10.6) входят три независимых 16-разрядных счетчика, работающих на вычитание в двоичном или двоично-десятичном коде. Сигналы тактового генератора или подсчитываемые сигналы из ВУ подаются на входы счетчиков CLK_0, CLK_1, CLK_2. Прием сигналов разрешается внешними управляющими сигналами $GATE_0$, $GATE_1$ и $GATE_2$ соответственно. Как только содержимое какого-либо счетчика становится равным нулю, вырабатывается один из выходных сигналов OUT_0, OUT_1 или OUT_3, которые могут использоваться как запросы на прерывание.

Инициализация каждого счетчика производится записью управляющего слова в регистр приказа. При этом устанавливаются начальное значение счетчика и один из нижеследующих шести режимов его работы — условий формирования выходного сигнала OUT.

Режим 0 — сигнал на выходе OUT формируется по окончании счета (программируемая задержка).

Режим 1 — программируемый одновибратор, т. е. формирование на выходе OUT сигнала длительностью N периодов сигнала CLK.

Режим 2 — генератор программируемой частоты, т. е. на выходе OUT формируется периодический сигнал с высоким уровнем в течение $N-1$ периодов сигнала CLK и с низким — в один период сигнала CLK.

Режим 3 — генератор прямоугольных импульсов со скважностью 2, т. е. на выходе OUT формируется сигнал высокого уровня в течение $N/2$ (N — четное) или $N+1/2$ (N — нечетное) периодов сигнала CLK.

Режим 4 — формирование одиночного строба с программным запуском, т. е. на выходе OUT формируется одиночный сигнал длительностью в один период сигнала CLK после приема N сигналов по входу CLK. Начало счета инициируется загрузкой в счетчик значения N.

Режим 5 — формирование одиночного строба с аппаратным запуском аналогично режиму 4, но начало счета инициируется фронтом сигнала GATE.

Обмен информацией между системным интерфейсом микроЭВМ и регистрами БИС осуществляется по шине данных $D_7,...,D_0$ с помощью управляющих сигналов $\overline{RD}$, $\overline{WR}$, $\overline{CS}$. При этом внутренние регистры БИС адресуются по линиям A_0 и A_1.

В качестве терминальных устройств в микроЭВМ наиболее широко используются алфавитно-цифровые дисплеи — видеотерминал плюс клавиатура (см. параграф 6.2). В составе МПК БИС серии КР580 имеются две специализированные БИС, предназначенные для построения дисплея, — программируемый контроллер электронно-лучевой трубки КР580ВГ75 и программируемый контроллер клавиатуры КР580ВВ79.

Многочисленные функции в микроЭВМ на основе МПК КР580, связанные с формированием системных управляющих сигналов, организацией и управлением работой системных шин и шин связи с ВУ, реализуются с использованием системных контроллеров и шинных формирователей КР580ВК28 и КР580ВК38, буферных регистров КР580ИР82 и КР580ИР83, шинного формирователя КР580ВА86 и генератора тактовых импульсов КР580ГФ24.

Заканчивая краткое описание МПК БИС серии КР580, отметим, что состав МПК постоянно расширяется за счет разработки новых типов БИС.

10.3. Микропроцессорный комплект К1810

Микропроцессорный комплект БИС серии К1810 является дальнейшим развитием комплекта БИС серии КР580. Входящий в состав серии К1810 16-разрядный микропроцессор КМ1810ВМ86 программно совместим с микропроцессором КР580ИК80А и аппаратно совместим со всеми БИС серии КР580. Такой подход к разработке комплекта БИС серии К1810 позволил использовать накопленный опыт разработки и программное обеспечение микроЭВМ на основе комплекта БИС серии КР580 при разработке более производительных микроЭВМ на основе БИС серии К1810.

В настоящее время микропроцессорный комплект БИС К1810 включает следующие БИС:

1) однокристальный 16-разрядный микропроцессор КМ1810ВМ86;
2) программируемый контроллер приоритетных прерываний КР1810ВН59;
3) системный контроллер КР1810ВК88;
4) арбитр системной шины КР1810ВБ89;
5) буферные регистры КР1810ИР82 и КР1810ИР83;
6) шинные формирователи КР1810ВА86 и КР1810ВА87;
7) генератор тактовых импульсов КР1810ГФ84.

Рассмотрим кратко структурную и функциональную организацию микропроцессора КМ1810ВМ86. Более подробно с микропроцессором КМ1810ВМ86 и другими БИС серии К1810, а также с принципами построения микроЭВМ и микропроцессорных систем на их основе можно ознакомиться в работах [16, 41].

Микропроцессор КМ1810ВМ86. Предназначен он для построения микропроцессорных систем с изолированными шинами системного интерфейса. Но так как в самом микропроцессоре реализована совмещенная шина «Адрес/Данные», то для формирования 20-разрядной шины адреса при использовании БИС КМ1810ВМ86 требуются внешние схемы. В микропроцессоре КМ1810ВМ86 реализован традиционный подход к выполнению команд, при котором производится последовательная выборка команд и данных из памяти микроЭВМ. Вместе с тем в микропроцессоре реализован ряд новых решений, существенным образом повышающих быстродействие и расширяющих его функциональные возможности:

1) развитая регистровая структура, включающая 14 программно-доступных 16-разрядных регистров, что значительно сокращает количество обращений к памяти;
2) конвейерный принцип выполнения команд с предварительной выборкой;
3) многофункциональное использование выводов БИС, что позволяет настраивать микропроцессор на различные применения;
4) формирование 20-разрядного адреса, что позволяет адресоваться к памяти емкостью до 1 Мбайт.

В структуре микропроцессора КМ1810ВМ86 (рис. 10.7 *а*) можно выделить устройства сопряжения с каналом, обработки, управления и синхронизации.

Устройство сопряжения с каналом обеспечивает: формирование 20-разрядного адреса памяти; выборку из памяти команд и операндов; формирование очереди команд; запись в память результатов выполнения команд. В устройство входят: шесть 8-разрядных регистров очереди команд; четыре 16-разрядных регистра для хранения адреса начала сегмента памяти; 16-разрядный регистр указателя адреса команды; 16-разрядный регистр обмена и 16-разрядный сумматор адреса.

Устройство обработки предназначено для выполнения операций, задаваемых командами программы. Устройство обработки не имеет непосредствен-

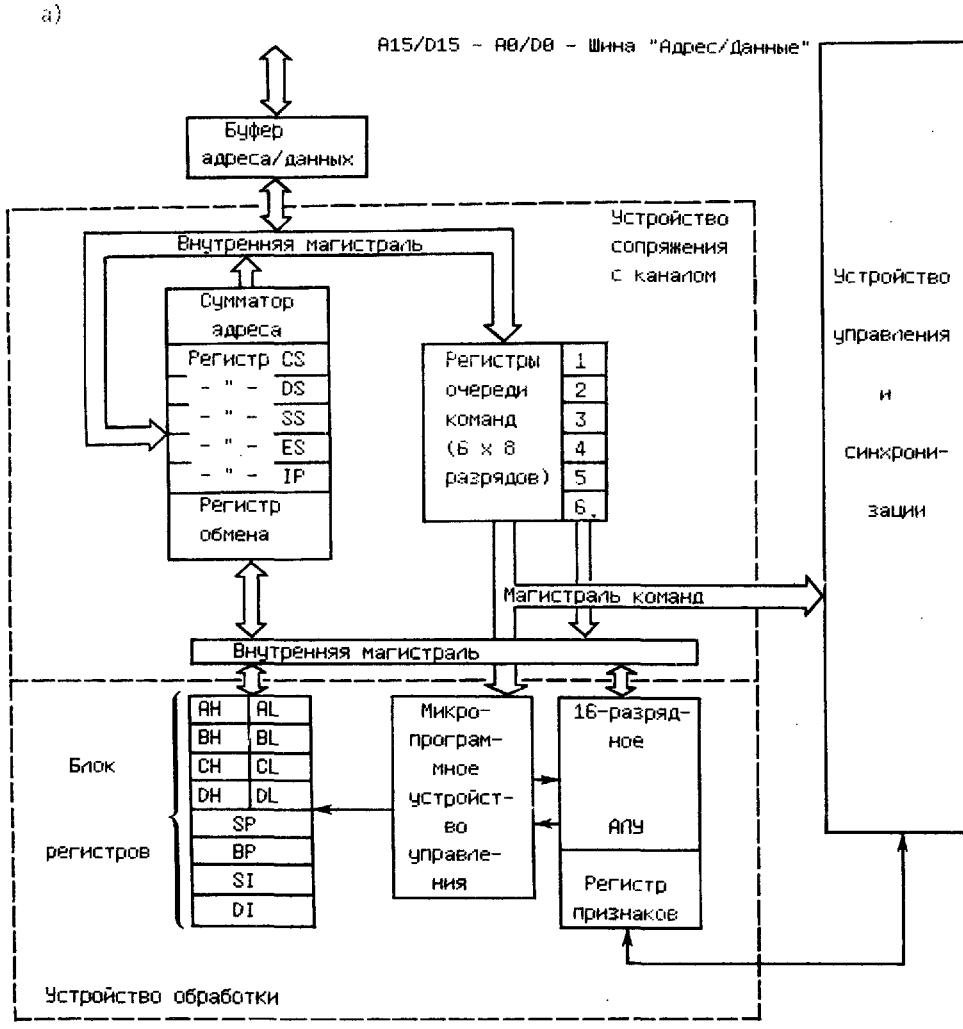


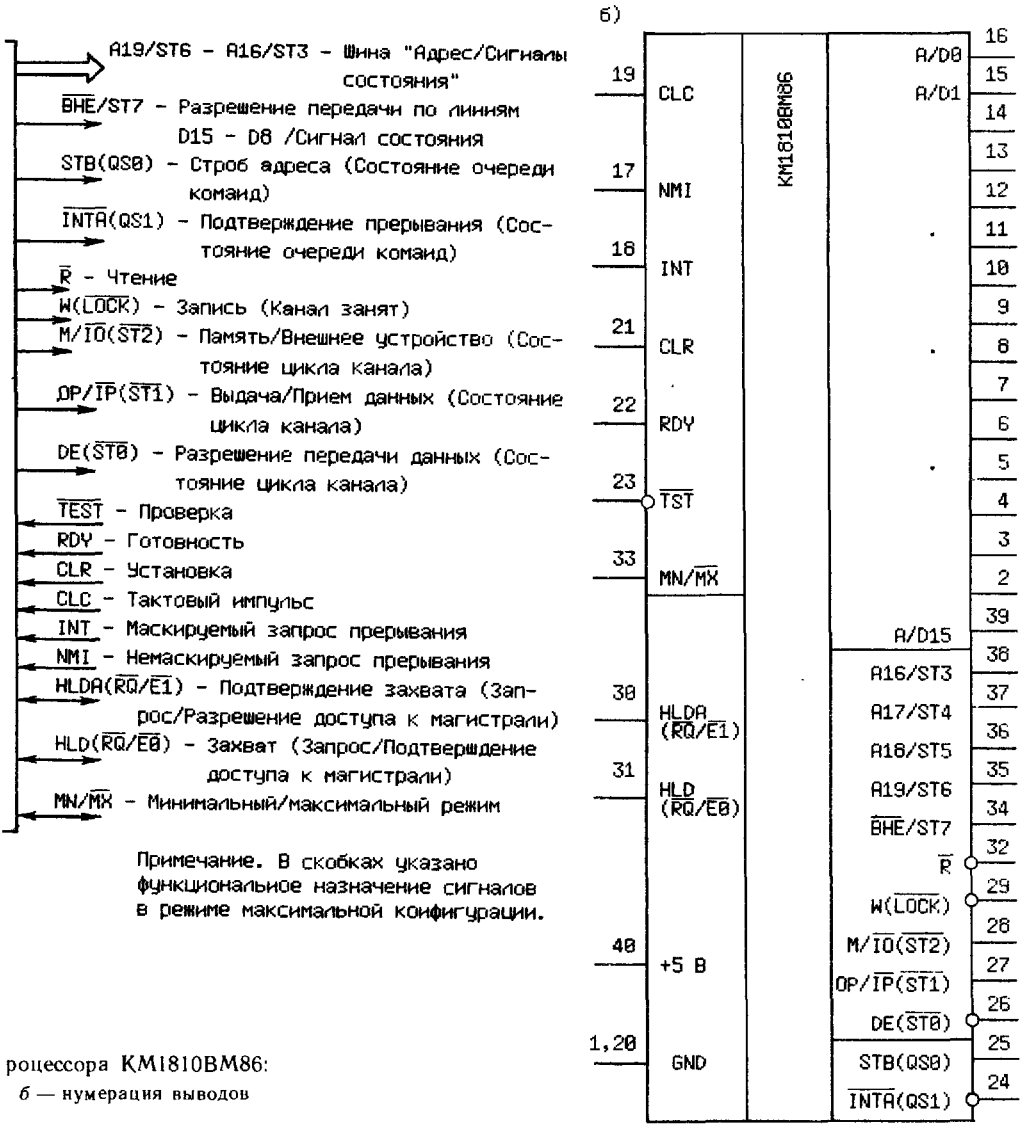
Рис. 10.7. БИС микроп
а — структурная схема;

ной связи с системным интерфейсом микроЭВМ. Команды в устройство обработки поступают из регистров очереди команд по 8-разрядной магистрали команд, а данные — через регистр обмена устройства сопряжения с каналом. В состав устройства обработки входят: 16-разрядное АЛУ; восемь 16-разрядных регистров общего назначения; регистр признаков состояния микропроцессора и микропрограммное устройство управления.

Часть регистров микропроцессора являются программно-доступными и имеют следующее функциональное назначение:

восемь регистров общего назначения (AX, BX, CX, DX, SP, BP, SI, DI) используются для хранения операндов, их адресов и результатов выполнения команд;

четыре сегментных регистра *CS*, *DS*, *SS* и *ES* предназначены для хранения базовых адресов текущих сегментов памяти;



роцессора КМ1810ВМ86:
б — нумерация выводов

регистр указателя (адреса) команд *IP* содержит адрес следующей выбираемой из памяти команды;

регистр признаков хранит признаки результатов выполнения команд и признаки управления прерываниями и режимами работы микропроцессора. [illegible] микропроцессора КМ1810ВМ86 обеспечивает выработку управляющих сигналов для системного интерфейса микроЭВМ на его основе в двух режимах функционирования: минимальной и максимальной конфигурациях.

Режим минимальной конфигурации (MN/$\overline{MX}$="1") применяется при использовании микропроцессора в простых однопроцессорных системах. В этом режиме микропроцессор вырабатывает все необходимые для функционирования микроЭВМ системные управляющие сигналы и обеспечивает режим прямого доступа к памяти по запросу контроллера прямого до-

ступа (например, БИС КР580ВТ57), используя управляющие сигналы HLD и HLDA.

Режим максимальной конфигурации ($MN/\overline{MX}$="0") применяется при использовании микропроцессора в сложных одно- и многопроцессорных системах. В этом режиме частично перестраивается структура устройства управления и синхронизации и изменяются функции восьми выводов микропроцессора (см. примечание на рис. 10.7, *а*).

Системный контроллер КР1810ВК88. Для управления системным интерфейсом в режиме максимальной конфигурации служит системный контроллер КР1810ВК88, который формирует расширенный набор управляющих сигналов. Системный контроллер принимает от микропроцессора сигналы состояния $\overline{ST0}$, $\overline{ST1}$ и $\overline{ST2}$, а остальные пять сигналов ($QS0$, $QS1$, $\overline{LOCK}$, $\overline{RQ/E0}$ и $\overline{RQ/E1}$) используются для организации многопроцессорных систем.

Система команд микропроцессора включает 135 команд, которые можно разделить на следующие группы: пересылки данных, арифметические, поразрядной обработки данных, обработки строк, передачи управления, управления процессором.

Команды пересылки обеспечивают передачу операнда-источника по адресу операнда-приемника. В эту группу входят команды ввода-вывода и команды перекодировки символов.

Арифметические команды обеспечивают выполнение основных арифметических операций над 8- и 16-разрядными двоичными целыми числами и с плавающей запятой, двоично-десятичными числами. В этой группе есть команды преобразования форматов данных — двоичного в двоично-десятичный и наоборот.

Команды поразрядной обработки предназначены для выполнения логических операций и операций сдвигов арифметических и циклических.

Команды обработки строк позволяют организовать для обработки строк данных длиной до 64 Кбайт аппаратные циклы, которые выполняются намного быстрее, чем соответствующие программы.

Команды передачи управления обеспечивают условные и безусловные переходы, управление циклами, вызовы подпрограмм и возвраты из подпрограмм. К ним относятся и команды управления прерываниями.

Команды управления процессом позволяют изменять содержимое некоторых разрядов регистра признаков и обеспечивают синхронизацию работы микропроцессора с другими микропроцессорами в многопроцессорных системах.

В заключение следует отметить, что на основе микропроцессора КМ1810ВМ86 с использованием БИС серий КР1810 и КР580 можно строить микропроцессорные системы различной вычислительной мощности и назначения от одноплатных микроконтроллеров до высокопроизводительных многопроцессорных систем.

10.4. Микропроцессорный комплект БИС К1801

Рассматриваемый комплект БИС — это специализированный комплект, предназначенный для построения различных моделей микроЭВМ с системой команд микроЭВМ «Электроника-60» (см. приложение В).

Основу комплекта составляют две БИС микропроцессора К1801ВМ1 и К1801ВМ2 и матричная БИС К1801ВП1, используемая для построения различных устройств управления и контроллеров ВУ. К настоящему времени на основе БИС К1801ВП1 реализованы следующие устройства:

1) контроллер управления ОЗУ емкостью 32К 16-разрядных слов на двух БИС — К1801ВП1-030 и К1801ВП1-013;

2) многофункциональные устройства параллельного обмена К1801ВП1-33 и К1801ВП1-034;

3) многофункциональное устройство последовательного обмена К1801ВП1-035.

В состав комплекта входит также ПЗУ К1801РЕ1 емкостью 4К 16-разрядных слов.

Сочетание однокристальных микропроцессоров и матричной БИС в одном комплекте создает хорошие предпосылки для построения широкого ряда программно-совместимых микроЭВМ различной конфигурации. Примером могут служить одноплатные микроЭВМ ряда «Электроника МС1201» (см. параграф 11.2).

На основе матричной БИС возможно быстрое изготовление различных специализированных и универсальных БИС, обеспечивающих как управление работой отдельных блоков микроЭВМ, так и взаимодействие с различными внешними устройствами микроЭВМ.

Рассмотрим кратко назначение отдельных БИС серии К1801. Более подробно с описаниями БИС серии К1801 и способами их применения можно ознакомиться в работе [20].

Микропроцессор К1801ВМ1. Этот конструктивно и функционально законченный модуль реализует систему команд микроЭВМ «Электроника-60». БИС К1801ВМ1 обеспечивает управление системным интерфейсом, аналогичным по составу управляющих сигналов и алгоритмам обмена системному интерфейсу микроЭВМ «Электроника-60».

В БИС К1801ВМ1 (рис. 10.8, *а*) можно выделить следующие функциональные блоки.

16-разрядный операционный блок обеспечивает формирование адресов команд и операндов, выполнение арифметических и логических операций, временное хранение операндов и результатов операций.

Блок микропрограммного управления предназначен для выработки последовательности микрокоманд, соответствующей выполняемой команде.

Блок прерываний обеспечивает работу приоритетной системы прерываний, производя прием и предварительную обработку внутренних и внешних запросов прерывания.

Интерфейсный блок обеспечивает обмен информацией между микропроцессором и устройствами, подключенными к системному интерфейсу.

Блок системной магистрали связывает внутреннюю магистраль микропроцессора с шиной «Адрес/Данные» системного интерфейса.

Схема тактирования обеспечивает синхронизацию всех функциональных блоков микропроцессора.

Функциональное назначение входных и выходных сигналов БИС К1801ВМ1 приведено на рис. 10.8, *а*, а нумерация выводов БИС — на рис. 10.8, *б*.

Микропроцессор К1801ВМ2. Являясь дальнейшим развитием БИС К1801ВМ1, он обладает расширенной системой команд и рядом новых функциональных возможностей. Расширение системы команд осуществляется за счет использования в БИС К1801ВМ2 (рис. 10.9, *а*) блока расширенной арифметики, который реализует команды умножения (MUL), деления (DIV), сдвига на N разрядов одиночного слова (ASH) и сдвига на N разрядов двойного слова (ASHC). Выполнение еще четырех команд, инициирующих операции над числами с плавающей запятой,— сложение (FADD), вычитание (FSUB), умножение (FMUL) и деление (FDIV) — обеспечивается на программном уровне с помощью системного ПЗУ, реализуемого на БИС К1801РЕ1.

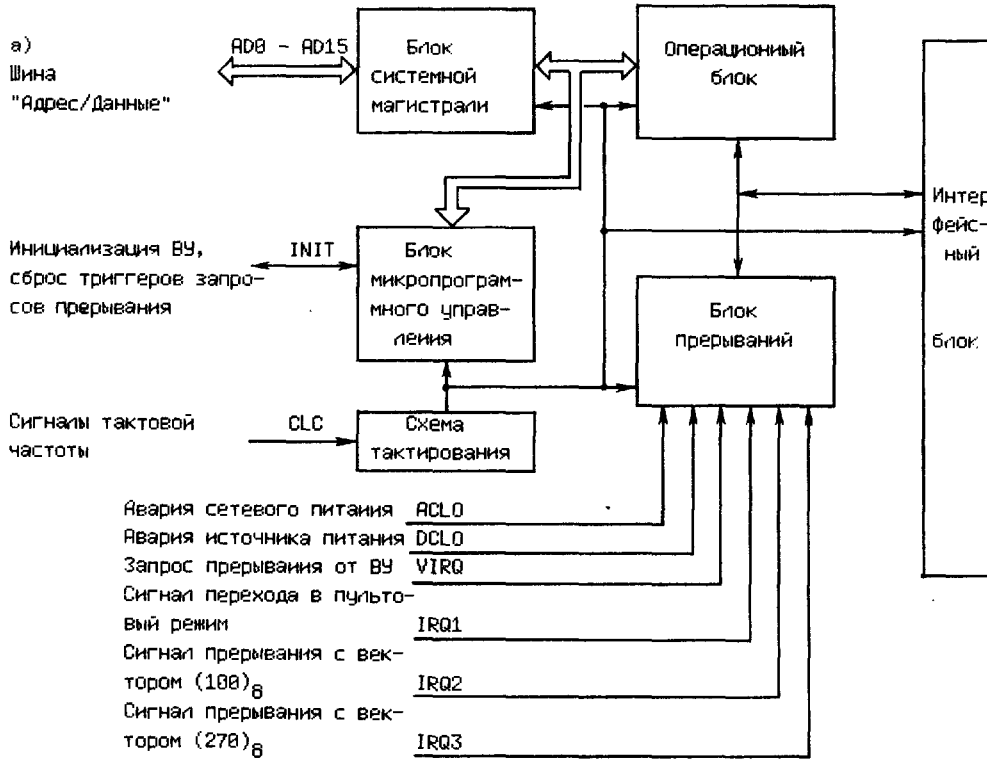


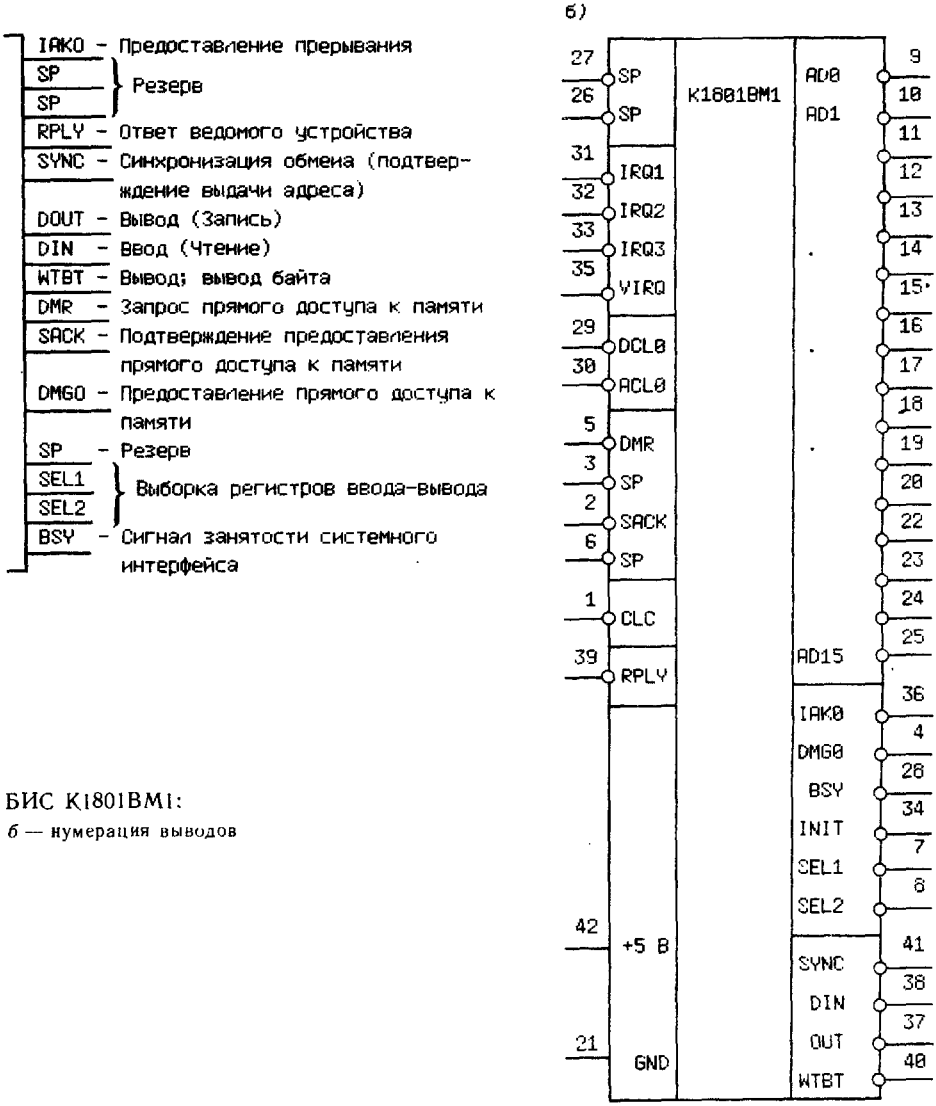
Рис. 10.8.
а — структурная схема;

Возможность работы с системной памятью как раз и является одной из новых функциональных возможностей микропроцессора К1801ВМ2. Для работы с системной памятью используются три управляющих сигнала: WRQ, WAKI и SEL. Управляющий сигнал SEL, выдаваемый одновременно с адресом, определяет обращение к дополнительной системной памяти и обеспечивает тем самым разделение обращений к системной и основной памяти микропроцессора. Таким образом, системная память не требует выделения специальных адресов и является как бы скрытой. Для системной памяти используется область адресного пространства с адреса 160000_8 по 163777_8, т. е. всего 2К адресов.

В микропроцессоре К1801ВМ2 реализованы два способа повышения скорости обработки данных.

Во-первых, реализован прием команд с опережением. Алгоритм приема и обработки команд построен таким образом, что к концу выполнения очередной команды следующая команда уже принята в буферный регистр команд и начинается выборка еще одной команды. Такой порядок выборки команд обеспечивает существенное повышение скорости обработки данных на линейных участках программы.

Во-вторых, реализован асинхронный режим передачи адреса при выполнении операций обмена по системному интерфейсу. Для этого используется управляющий сигнал «Адрес принят» (AR), который вырабатывается ВУ (пассивным устройством) после приема адреса. Тем самым сокращается время выполнения каждого цикла обмена по системному интерфейсу микро-ЭВМ на основе БИС К1801ВМ2.



БИС К1801ВМ1:
б — нумерация выводов

Функциональное назначение входных и выходных сигналов микропроцессора К1801ВМ2 приведено на рис. 10.9, *а*, а нумерация выводов — на рис. 10.9, *б*.

Матричная БИС К1801ВП1. Она представляет собой базовый кристалл, содержащий матрицу стандартных логических элементов, которые могут быть соединены между собой одним или несколькими слоями металлизации (см. параграф 5.5).

С помощью слоев металлизации реализуются требуемые схемы соединений логических элементов в матрице, определяющие конкретное ее назначение. Выше были перечислены некоторые устройства, реализуемые на основе БИС К1801ВП1.

Здесь в качестве примера использования матричной БИС рассмотрим назначение и структурные схемы устройств, реализуемых только на одной БИС — К1801ВП1-034.

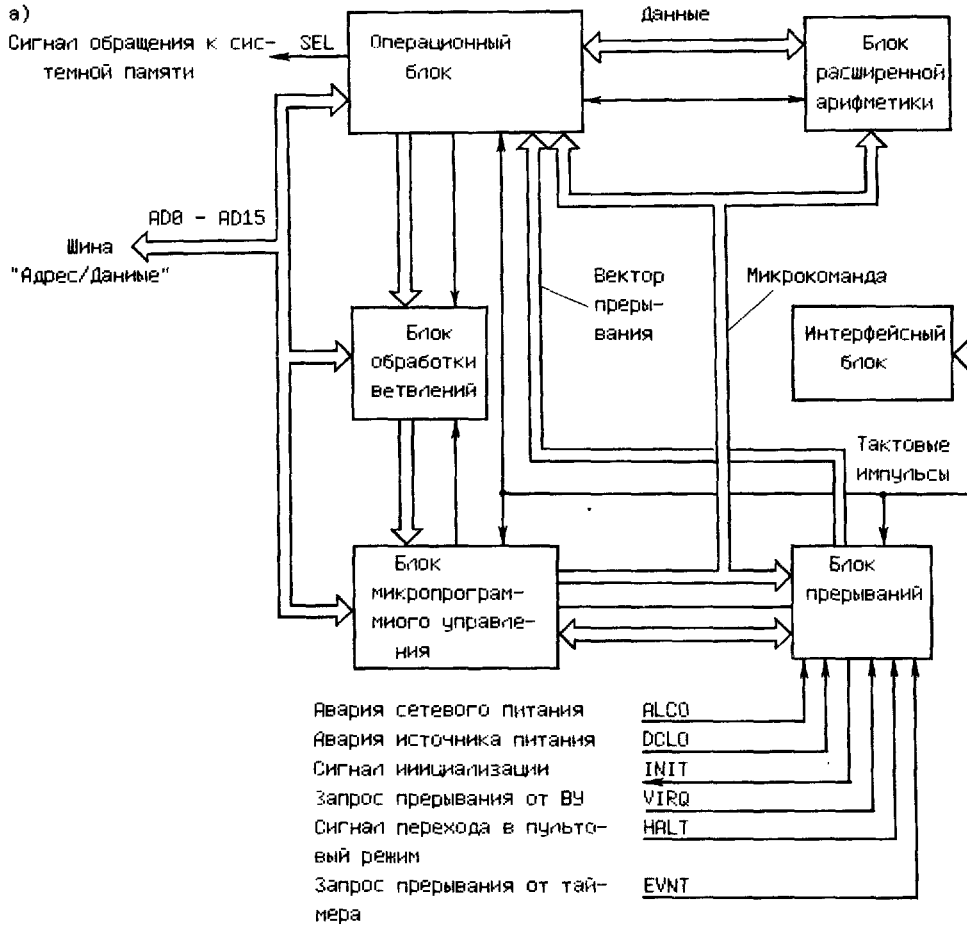


Рис. 10.9. БИС
а - структурная схема;

Матричная БИС К1801ВП1-34. Это многофункциональное устройство может работать в следующих режимах:

передачи информации (рис. 10.10);

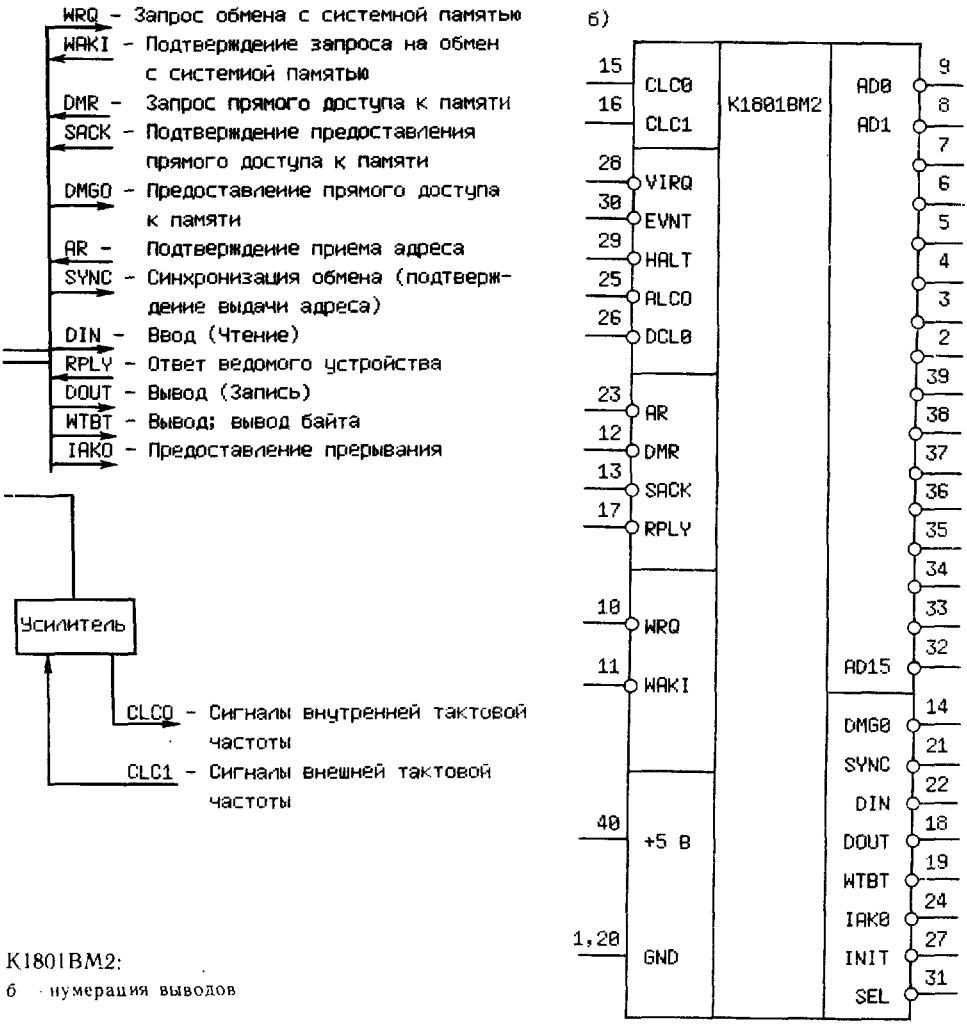
буферного регистра данных (рис. 10.11);

выдачи вектора прерывания и компаратора адреса (рис. 10.12).

Установка БИС К1801ВП1-034 в соответствующий режим работы производится с помощью управляющих сигналов RC0 и RC1 (рис. 10.10, *а*, 10.11, *а*, 10.12, *а*).

Структурная схема и функциональное назначение выводов БИС К1801ВП1-034 в режиме устройства передачи информации представлены на рис. 10.10, *а*.

С помощью управляющих сигналов $\overline{CA}$ и $\overline{CB}$ данные с входных шин *A0*, ..., *A7* и *B0*, ..., *B7* соответственно передаются на двунаправленную выходную шину *C0*, ..., *C7* в прямом или инверсном коде в зависимости от управляющего сигнала COM. Данные из блока ввода-вывода БИС могут быть записаны в буферный регистр и затем выданы в шину *D0*, ..., *D7* в прямом или инверсном коде в зависимости от управляющего сигнала COM.



К1801ВМ2:
б – нумерация выводов

Нумерация выводов БИС К1801ВП1-034 в режиме передачи информации приведена на рис. 10.10,*б*.

Структурная схема буферного регистра данных на основе матричной БИС К1801ВП1-034 и функциональное назначение выводов БИС представлены на рис. 10.11, *а*.

Информация с входной шины *D0*, ..., *D15* по управляющему сигналу С записывается в 16-разрядный буферный регистр. Сигнал DMF разрешает выдачу данных из буферного регистра на выходную шину $\overline{AD0}$,...,$\overline{AD15}$. Нумерация выводов БИС К1801ВП1-034 в режиме буферного регистра данных приведена на рис. 10.11, *б*.

Устройство выдачи вектора прерывания и компаратора адреса на основе БИС К1801ВП1-034 (рис. 10.12,*а*) используются в контроллерах ВУ для формирования адреса вектора прерывания, значение которого устанавливается на входах *S11*,...,*S16*, и селекции адреса ВУ, значение которого устанавливается на входах *S1*,...,*S10*. Значение адреса ВУ сравнивается с адресом, поступающим по шине *AD3*, ..., *AD12*, и при их равенстве вырабатывается сигнал $\overline{SB}$ («Устройство выбрано»). Таким образом, на одной

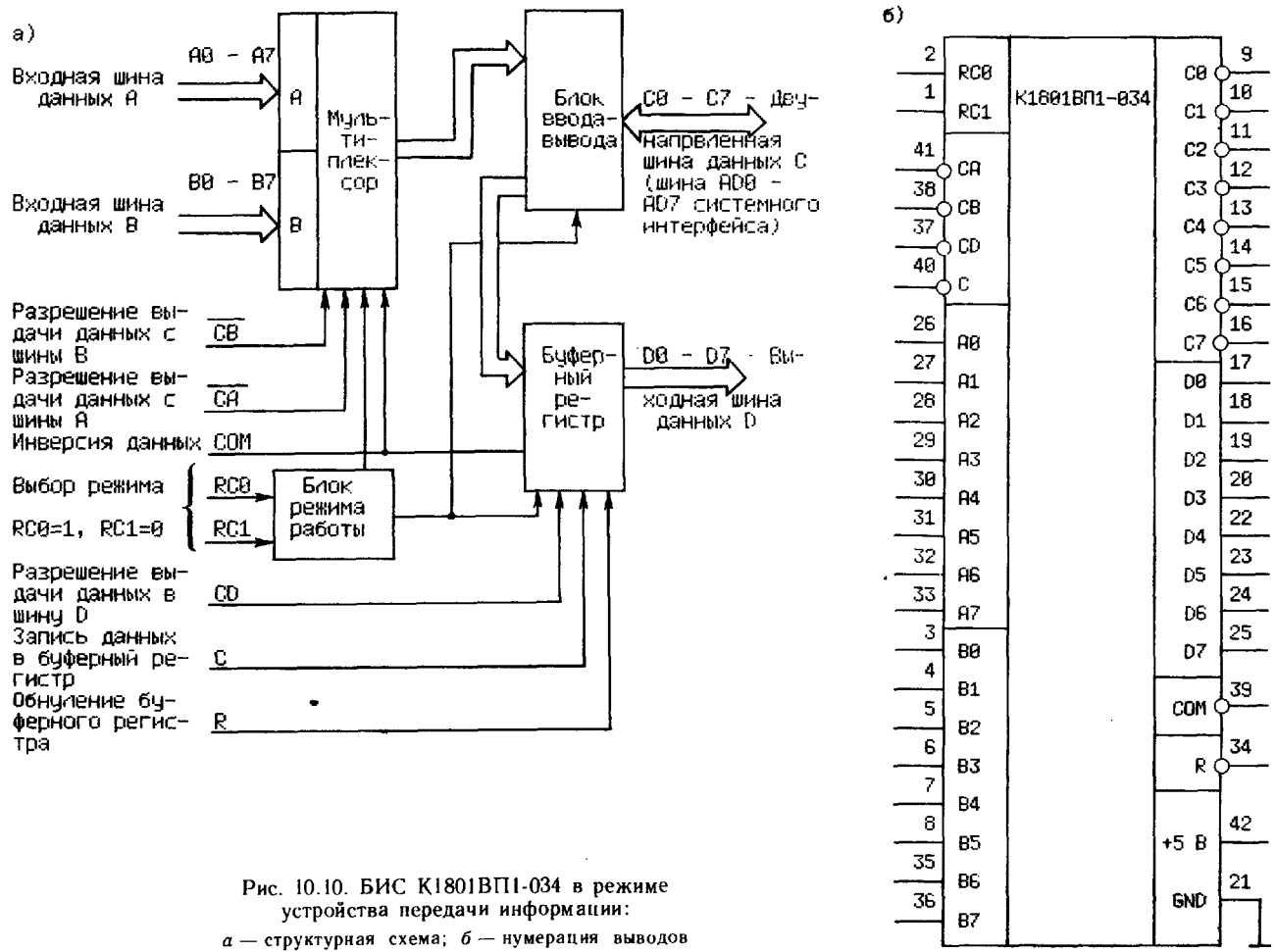
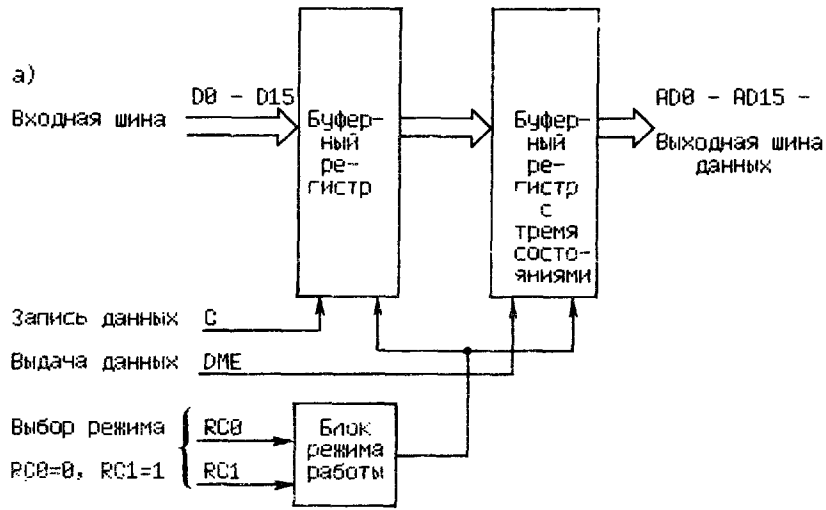


Рис. 10.10. БИС К1801ВП1-034 в режиме устройства передачи информации:

а — структурная схема; *б* — нумерация выводов



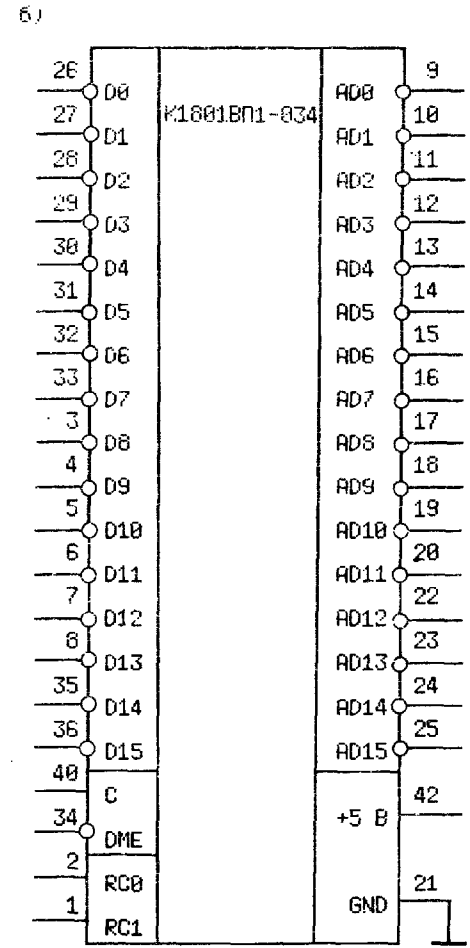
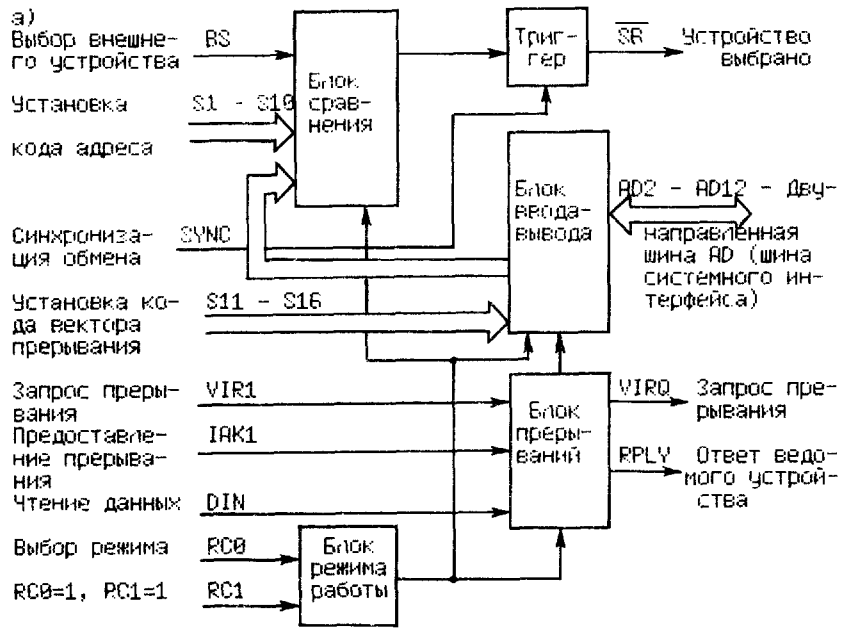


Рис. 10.11. БИС К1801ВП1-034 в режиме буферного регистра данных:

а – структурная схема; *б* – нумерация выводов



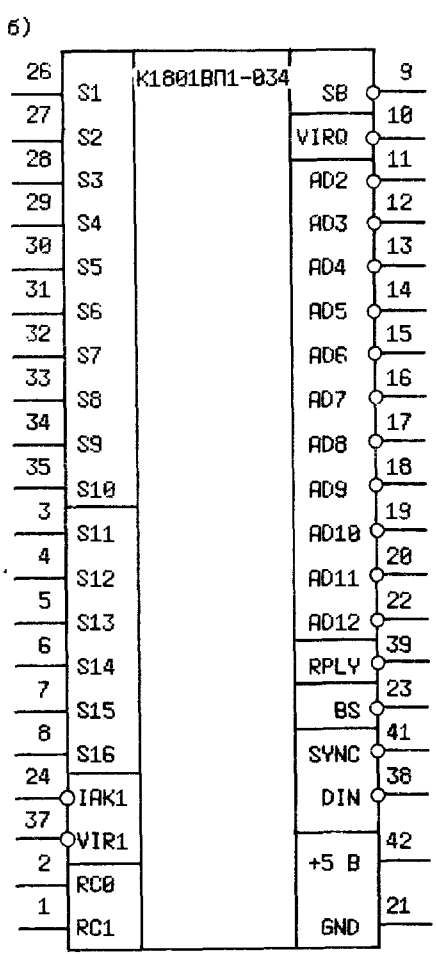


Рис. 10.12. БИС К1801ВП1-034 в режиме устройства выдачи вектора прерывания и компаратора адреса:

а — структурная схема; *б* — нумерация выводов

БИС К1801ВП1-034 в контроллере ВУ реализуются логика выбора ВУ и выдача вектора прерывания. Нумерация выводов БИС К1801ВП1-34 в рассмотренном режиме приведена на рис. 10.12,*б*.

Укажем еще несколько применений БИС К1801ВП1:

контроллер накопителя на гибких магнитных дисках на основе БИС К1801ВП1-033;

стандартный параллельный интерфейс (ИРПР) на БИС К1801ВП1-033 и К1801ВП1-034;

стандартный последовательный интерфейс (ИРПС) на БИС К1801ВП1-035.

В заключение следует отметить, что микропроцессорный комплект БИС К1801 входит в ряд МПК БИС, предназначенных для построения микроконтроллеров, микроЭВМ и микропроцессорных систем, программно-совместимых с семейством ЭВМ СМ3 и СМ4 ряда СМ ЭВМ. К указанному ряду МПК БИС следует отнести БИС серий К581, К1801, К1809 и К1811.

10.5. Микропроцессорный комплект БИС К1804

БИС микропроцессорного комплекта серии К1804 предназначены для построения микропроцессорных устройств (микроЭВМ и микроконтроллеров) с разрядно-модульной организацией. Большинство БИС серии К1804 — четырехразрядные микропроцессорные секции, используемые либо для обработки четырех разрядов данных, либо для выполнения определенного набора управляющих функций. Секционность БИС комплекта обеспечивает возможность построения на их основе микропроцессорных устройств с разрядностью, кратной четырем при параллельном включении БИС. Управление работой отдельных БИС комплекта осуществляется микропрограммно, что позволяет с помощью БИС серии К1804 реализовать практически любой набор команд.

В настоящее время в состав комплекта входят девять БИС, которые можно разделить на две группы: БИС для построения операционной части микропроцессора и БИС для построения управляющей части микропроцессора.

В первую группу входят:

1) микропроцессорная секция К1804ВС1;
2) микропроцессорная секция К1804ВС2;
3) схема ускоренного переноса К1804ВР1;
4) четырехразрядный параллельный регистр К1804ИР1;
5) схема управления состоянием и сдвигами К1804ВР2.

Вторую группу составляют:

6) секции управления адресом микрокоманды К1804ВУ1 и К1804ВУ2;
7) схема управления следующим адресом К1804ВУ3;
8) схема управления последовательностью микрокоманд К1804ВУ4.

Рассмотрим кратко назначение и структурные схемы отдельных БИС серии К1804. Подробное описание МПК БИС серии К1804 и принципов проектирования микропроцессорных устройств на их основе приведено в работе [33].

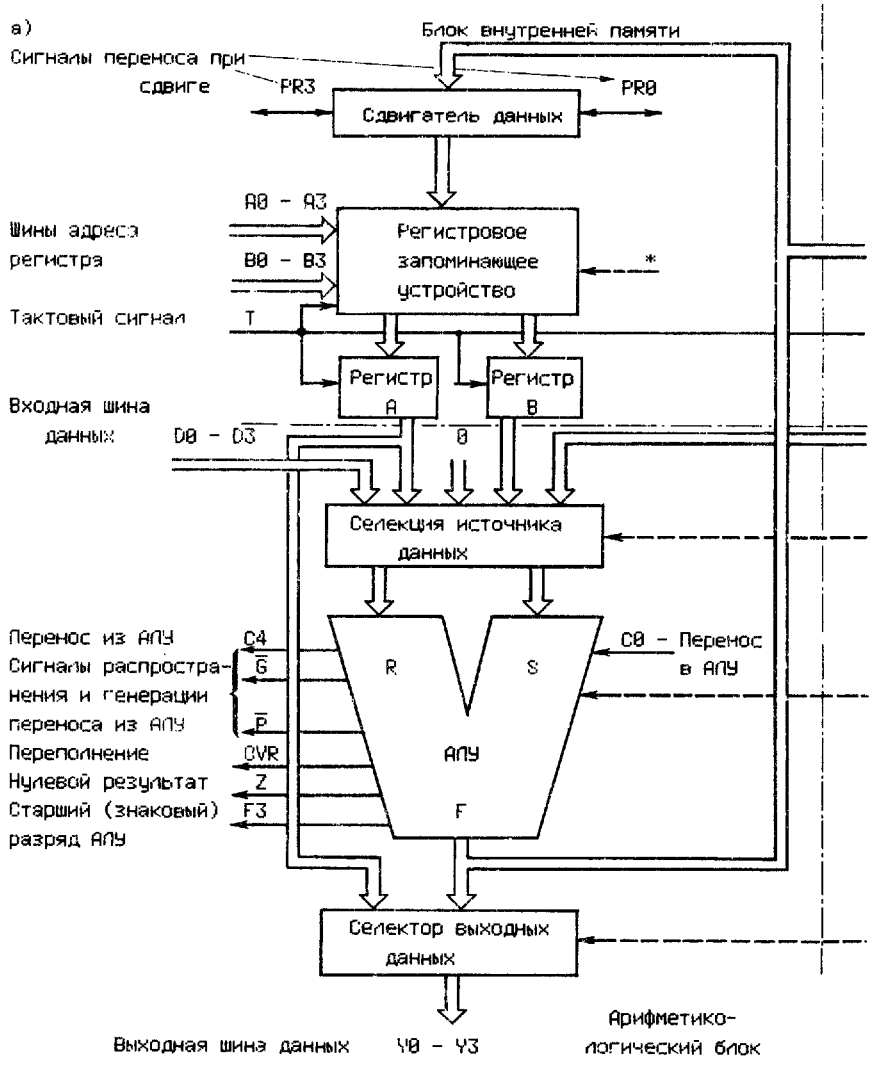
Микропроцессорная секция К1804ВС1 (рис. 10.13). В состав этого четырехразрядного устройства для обработки данных входят следующие блоки: арифметико-логический, внутренней памяти, регистра *Q* и управления.

АЛУ, входящее в состав **арифметико-логического блока** БИС К1804ВС1, выполняет восемь арифметических и логических операций над двумя операндами, подаваемыми на входы *R* и *S* через селектор источника данных. АЛУ формирует четыре признака результата: перенос из старшего разряда *C4*, переполнение *OVR*, знак (содержимое старшего разряда) *F3* и признак

нулевого результата *Z*. Источниками данных (операндов) могут быть: входная шина данных, содержимое одного или двух внутренних регистров БИС, содержимое регистра *Q* или нулевая константа. Выбор операндов и требуемой операции АЛУ производится с помощью разрядов микрокоманды *I0*, ..., *I2* и *I3*, ..., *I5* соответственно. Перечень операций АЛУ приведен в табл. 10.1. Результат операции выдается на выходную шину данных через селектор данных, управляемый разрядами *I6*, ..., *I8* микрокоманды.

Сигналы выходного (*C4*) и входного (*C0*) переносов АЛУ обеспечивают возможность комплексирования БИС в блоки с разрядностью, кратной четырем. Ускоренный перенос при комплексировании БИС организуется с помощью внешних схем с использованием сигналов $\overline{P}$ и $\overline{G}$ АЛУ.

Блок внутренней памяти БИС содержит 16 четырехразрядных регистров и имеет два независимых канала адресации — шины адреса А и В. Запись в регистры возможна только при адресации по шине В. Информация поступает в регистры с выхода АЛУ через сдвигатель данных, который управ-



ляется разрядами *16*, ..., *18* микрокоманды и обеспечивает три режима записи: без сдвига, со сдвигом на один разряд влево или вправо. При сдвиге влево (вправо) формируется сигнал переноса PR3 (PR0).

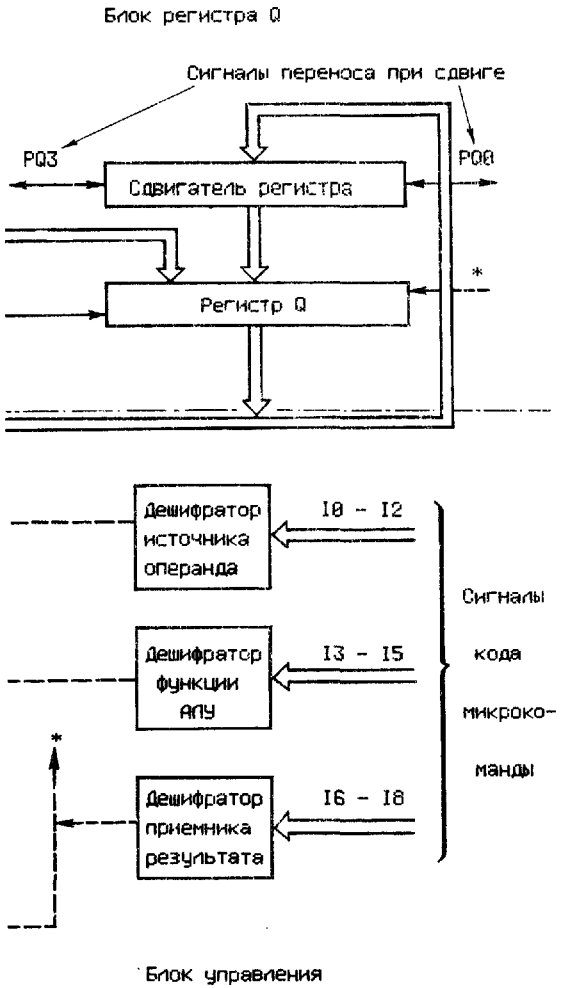
Информация с выхода АЛУ может быть также записана в дополнительный рабочий регистр Q. Содержимое регистра Q можно сдвинуть на один разряд влево или вправо. Управление записью в регистр Q и сдвигами его содержимого осуществляется разрядами *16*, ..., *18* микрокоманды.

Таблица 10.1

Перечень операций АЛУ

Значения разрядов микрокоманды			Операция АЛУ
15	*14*	*13*	
0	0	0	$R+S+CO$
0	0	1	$S-R-1+CO$
0	1	0	$R-S-1+CO$
0	1	1	$R \vee S$
1	0	0	$R \wedge S$
1	0	1	$\overline{R} \wedge S$
1	1	0	$R \oplus S$
1	1	1	$\overline{R} \oplus S$

Все внутренние управляющие сигналы в БИС формируются блоком управления из сигналов кода микрокоманды, поступающих по шине микрокоманды *10*, ..., *18* из микропрограммной памяти.



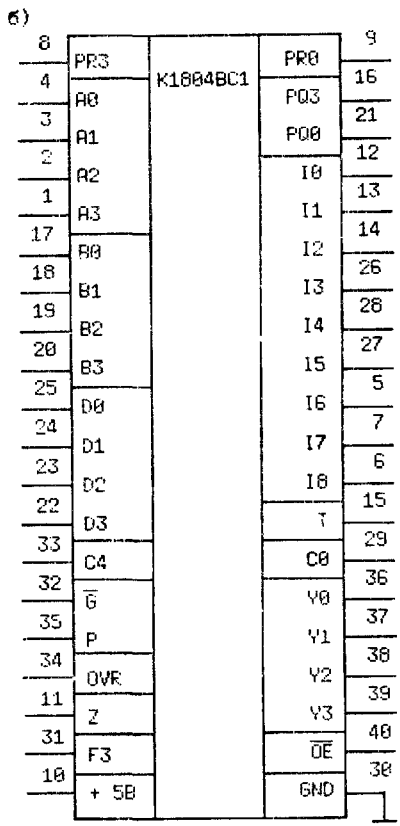
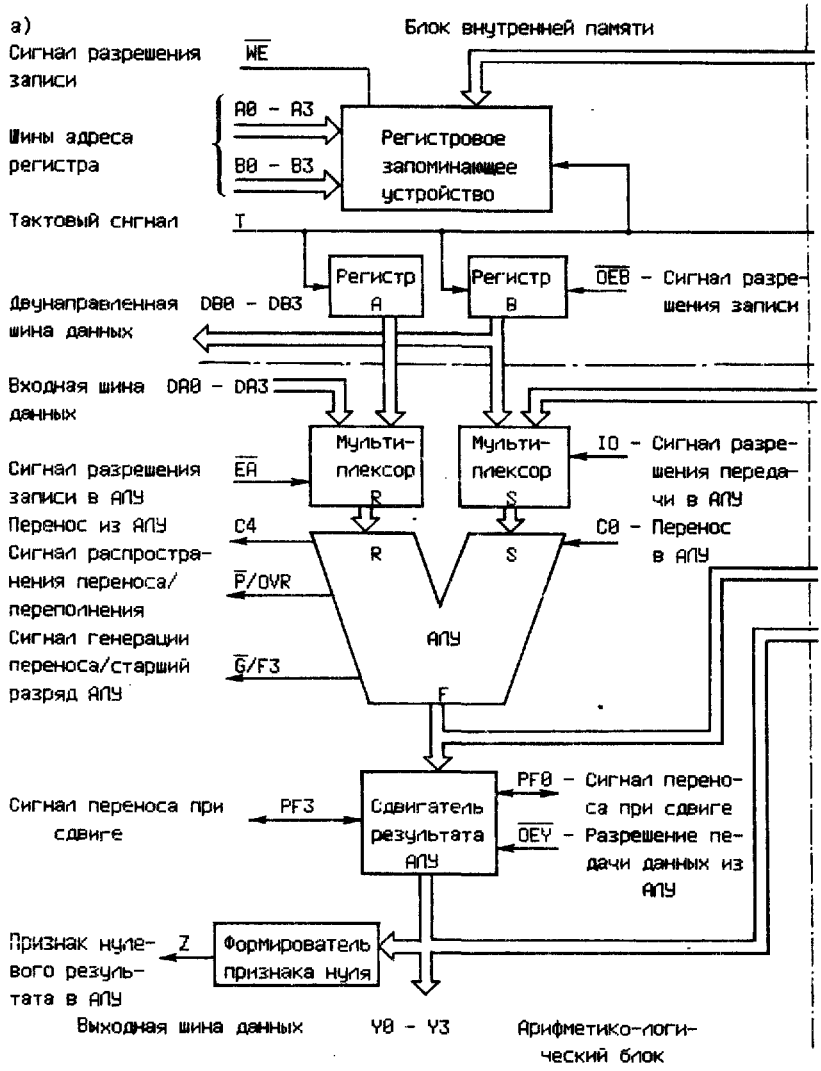


Рис. 10.13. БИС K1804BC1:

а — структурная схема; *б* — нумерация выводов



Микропроцессорная секция К1804ВС2 (рис. 10.14). Это четырехразрядное устройство для обработки данных состоит из тех же функциональных узлов, что и БИС К1804ВС1. Однако функциональные возможности БИС К1804ВС2 намного шире.

Арифметико-логический блок БИС, в который входят АЛУ, сдвигатель результата АЛУ и формирователь признака нуля, обеспечивает выполнение семи арифметических, девяти логических и девяти специальных функций над одним или двумя операндами, поступающими на вход АЛУ через входные мультиплексоры MR и MS. С использованием сдвигателя результата АЛУ, внутренней регистровой памяти и рабочего регистра Q в БИС выполняются арифметические и логические сдвиги, микропрограммы умножения, деления, нормализации и преобразования чисел со знаком в дополнительный код. В БИС выполняются и операции расширения знака числа, генерации паритета на выходе PF0 (операции «Исключающее ИЛИ» всех выходов АЛУ и сигнала на входе PF3), увеличения операнда на 1 (2).

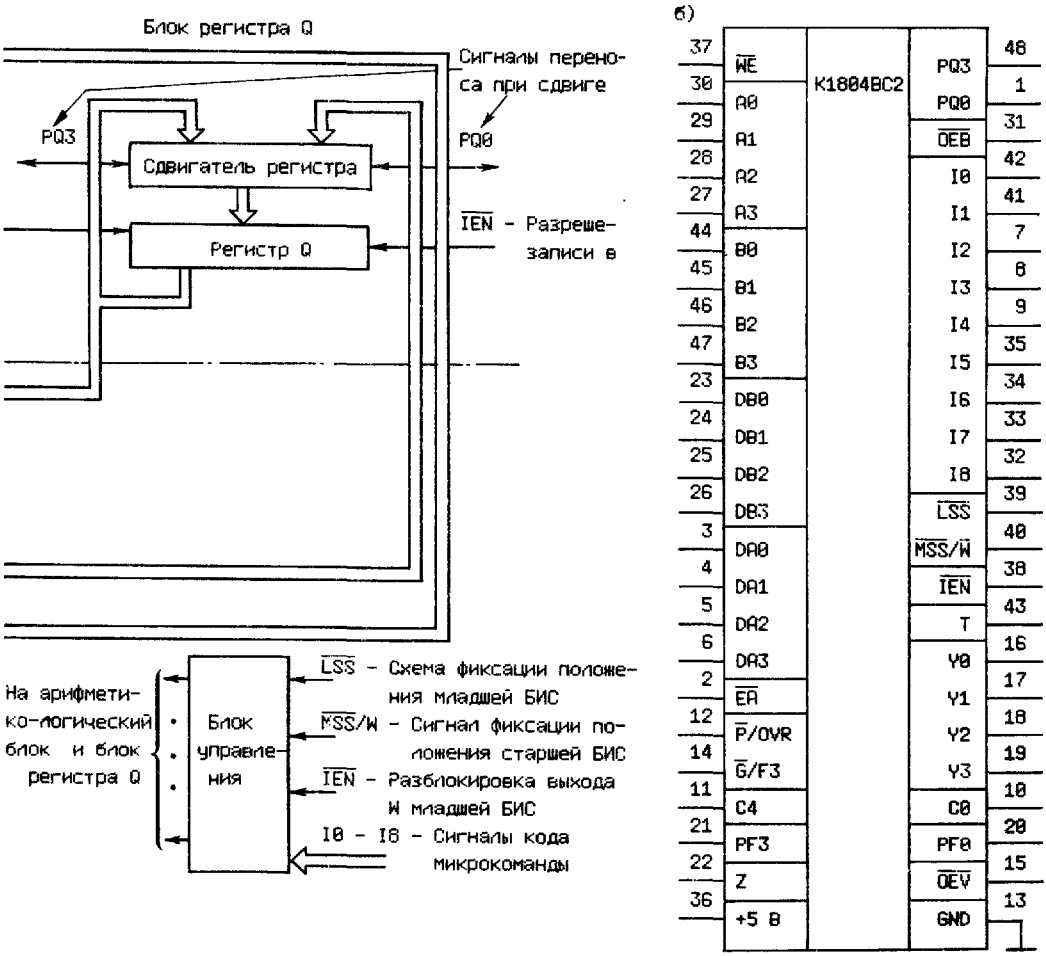


Рис. 10.14. БИС К1804ВС2:
а — структурная схема; *б* — нумерация выводов

При комплексировании микропроцессорных секций К1804ВС2 возникает необходимость их настройки на заданное положение в системе (старшая, младшая или средняя). Это связано с тем, что функционирование БИС при выполнении некоторых операций зависит от их положения в системе. Настройка БИС выполняется с помощью сигналов $\overline{LSS}$ и $\overline{MSS}/\overline{W}$.

Блок внутренней памяти БИС К1804ВС2 по количеству внутренних регистров и их адресации аналогичен блоку внутренней памяти БИС К1804ВС1. Но вместе с тем он обладает новыми функциональными возможностями — способен работать в режимах двух- и трехадресной обработки. При двухадресной обработке на входы A0,...,A3 подается адрес операнда R, а на входы B0,...,B3 — адрес операнда S, который одновременно является и адресом результата. При трехадресной обработке адрес на входах B0,...,B3 после считывания операнда S заменяется на адрес результата в течение одного такта и тем самым обеспечивается запись результата операции АЛУ по новому адресу.

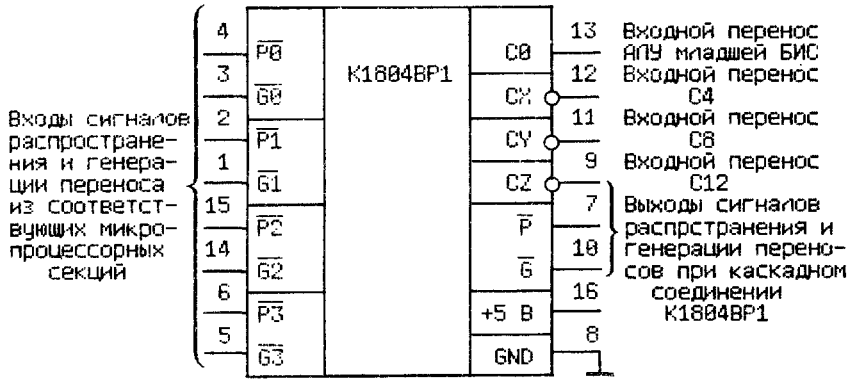


Рис. 10.15. Функциональное назначение и нумерация выводов схемы ускоренного переноса К1804ВР1

Блок управления микропроцессорной секции обеспечивает формирование всех внутренних управляющих сигналов БИС, определяющих выполняемую операцию АЛУ и положение БИС в системе. Входными сигналами блока управления являются сигналы кода микрокоманды *I0*, ..., *I8* и системные управляющие сигналы $\overline{LSS}$, $MSS/\overline{W}$ и $\overline{IEN}$.

Нумерация выводов БИС К1804ВС2 приведена на рис. 10.14, *б*.

Схема ускоренного переноса К1804ВР1 (рис. 10.15). Используется она для организации параллельных цепей переноса в блоках обработки данных на основе БИС К1804ВС1 и К1804ВС2 с разрядностью не более 16. При большей разрядности может быть применено каскадное соединение схем ускоренного переноса К1804ВР1.

Четырехразрядный параллельный регистр К1804ИР1 (рис. 10.16). Предназначен он для временного хранения 4-разрядного слова данных и последующей его выдачи по двум независимым шинам данных. Регистр К1804ИР1 может служить в качестве буферного устройства между двумя шинами.

Схема управления состоянием и сдвигами К1804ВР2 (рис. 10.17). Предназначена она для выполнения различных операций по обслуживанию АЛУ:

формирования сигналов входного переноса для микропроцессорных секций и схем ускоренного переноса;

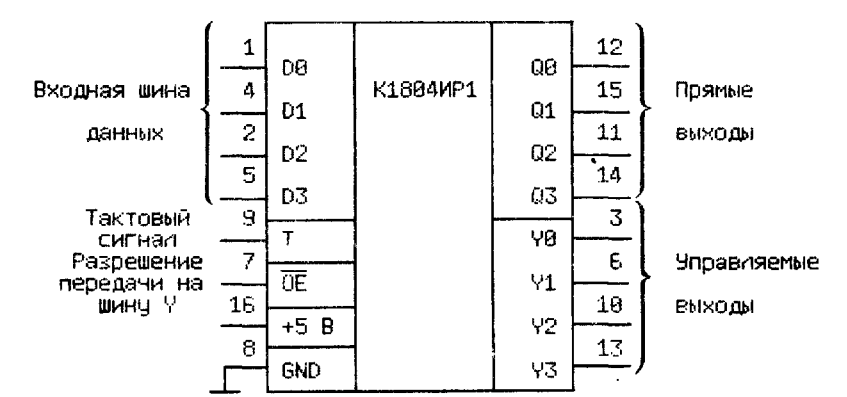


Рис. 10.16. Функциональное назначение и нумерация выводов регистра К1804ИР1

организации арифметических, логических и циклических сдвигов чисел одинарной и двойной длины (всего 32 варианта сдвигов);

выполнения различных операций с содержимым двух внутренних регистров состояния *M* и *N*;

проверки за один такт одного из 16 различных условий, поступающих с выходов регистров состояния *M* и *N* либо из микропроцессорной секции.

В БИС К1804ВР2 можно выделить следующие функциональные блоки.

Блок обработки признаков предназначен для хранения и модификации следующих признаков выполнения операций в АЛУ микропроцессорной секции: перенос (C), знак (N), переполнение (V) и нулевой результат (Z).

Блок проверки условия обеспечивает проверку 16 различных условий (функций признаков состояния АЛУ) и формирование кода условия (0 или 1).

Блок управления переносом формирует сигнал входного переноса для АЛУ микропроцессорной секции из семи различных входных сигналов, что позволяет легко реализовать операции сложения и вычитания чисел одинарной и двойной длины.

Блок управления сдвигами обеспечивает реализацию 32 различных вариантов арифметических, логических и циклических сдвигов.

Управляющие сигналы для всех блоков БИС К1804ВР2 формируются блоком управления из сигналов I0, ..., I12 кода микрокоманды:

БИС К1804ВР2 используется совместно с микропроцессорными секциями К1804ВС1 и К1804ВС2, а также со схемой управления последовательностью микрокоманд К1804ВУ4, которая будет рассмотрена дальше.

Секции управления адресом микрокоманды К1804ВУ1 и К1804ВУ2. Четырехразрядные секции К1804ВУ1 и К1804ВУ2 используются для построения блоков микропрограммного управления различных цифровых устройств. Основной функцией этих БИС является формирование адреса микрокоманды под воздействием внешних управляющих сигналов. В БИС К1804ВУ1 реализованы три способа адресации микрокоманд: последовательная по счетчику микрокоманд; выборка адреса из одного из внутренних или внешних источников адреса; модификация младших разрядов адреса микрокоманды. В БИС К1804ВУ2 реализованы только два первых способа адресации. Обе БИС обеспечивают реализацию шести различных управляющих конструкций в алгоритмах микропрограммного управления:

безусловный переход;

условный переход по одному или нескольким направлениям;

цикл с проверкой условия;

повторение предыдущего адреса микрокоманды;

условный переход к микропрограмме и возврат по условию;

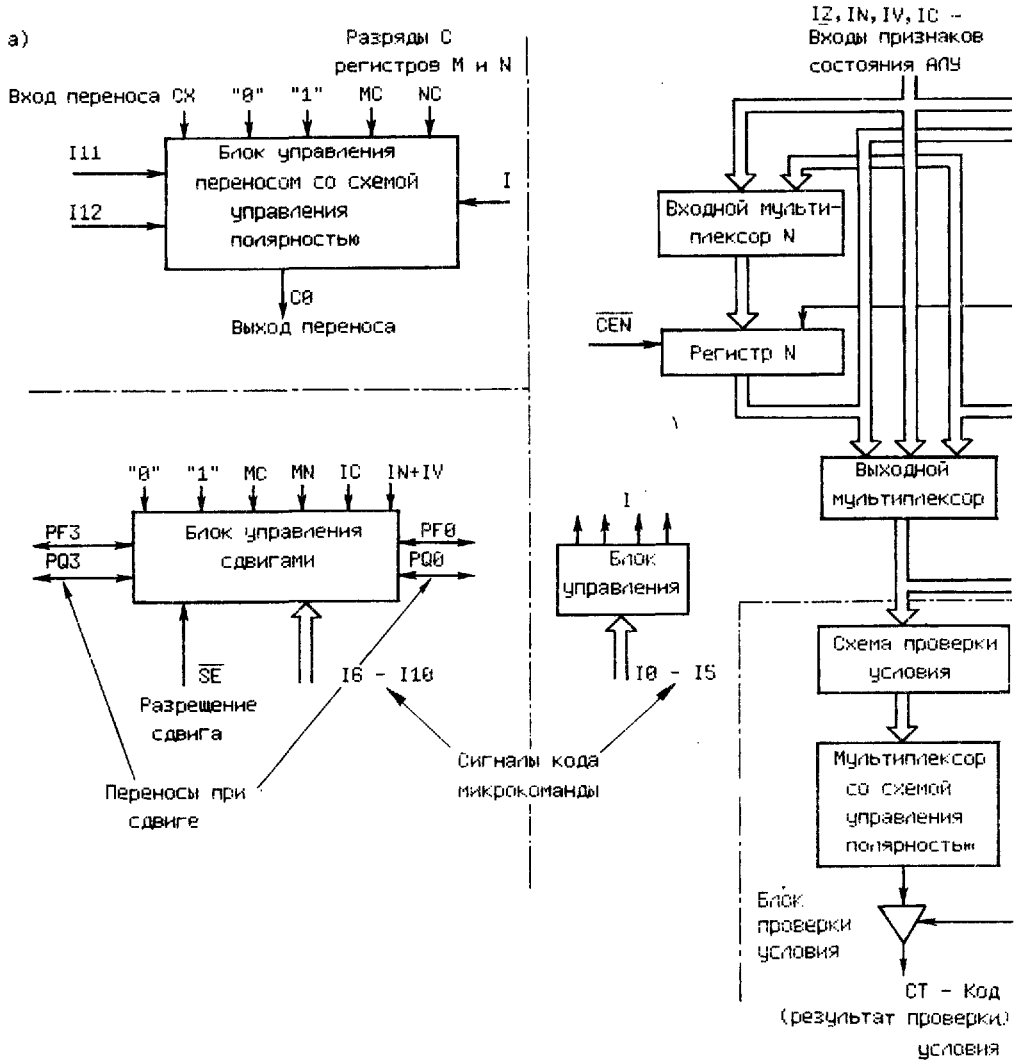
безусловный переход к микропрограмме и возврат.

Реализация в БИС К1804ВУ1 и К1804ВУ2 перечисленных выше способов адресации и управляющих конструкций обеспечивается с помощью следующих функциональных блоков, входящих в состав БИС (рис. 10.18).

Блок выборки адреса содержит мультиплексор, с помощью которого в зависимости от управляющих сигналов *S0* и *S1*, выбирается один из четырех источников адреса: регистр адреса, стек, счетчик микрокоманд или входная шина адреса *D0*,...,*D3*. Выбранный адрес может быть модифицирован с помощью маски, подаваемой по шине QR0, ..., QR3.

Регистр адреса используется для хранения адреса, принятого от внешнего источника по шинам *R0*,...,*R3*.

Счетчик микрокоманд обеспечивает увеличение на единицу выбранного адреса и передачу его либо в стек, либо снова в блок выборки адреса.



Стек емкостью четыре 4-разрядных слова используется для временного хранения выбранных адресов с целью их последующего использования в различных управляющих конструкциях алгоритмов микропрограммного управления.

Необходимая разрядность (кратная четырем) адреса микропрограммной памяти обеспечивается параллельным включением соответствующего количества БИС управления адресом микрокоманды.

На рис. 10.18,*б* приведена нумерация выводов БИС К1804ВУ1 и К1804ВУ2. Сокращение количества выводов в БИС К1804ВУ2 достигнуто за счет того, что входы адреса *D0*, ..., *D3* используются как входы регистра адреса, а входы маски *OR0*, ..., *OR3* и входы схемы ИЛИ отсутствуют.

Схема управления следующим адресом К1804ВУ3 (рис. 10.19). Предназначена она для преобразования закодированного поля микрокоманды в набор управляющих сигналов для различных узлов блока микропрограммного управления микропроцессорной системы. Такими узлами могут быть

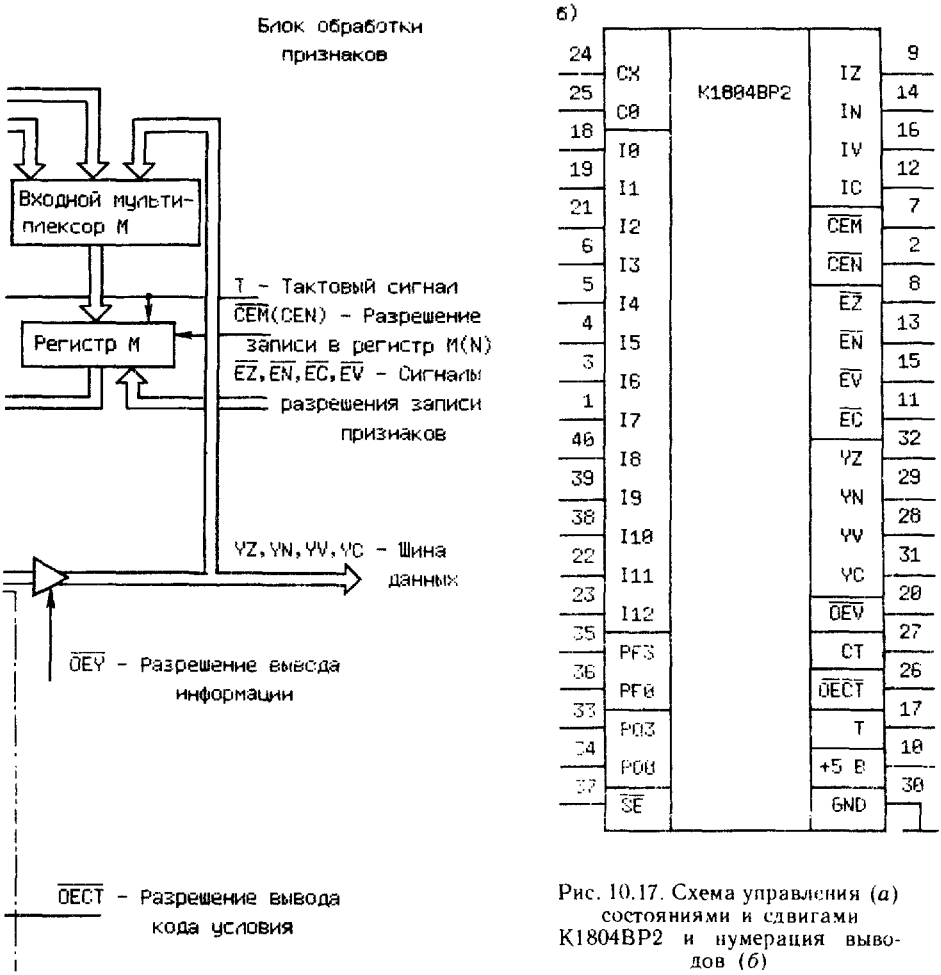


Рис. 10.17. Схема управления (*а*) состояниями и сдвигами К1804ВР2 и нумерация выводов (*б*)

секции управления адресом микрокоманды К1804ВУ1 и К1804ВУ2, счетчик циклов, регистр микрокоманд и т. д.

БИС К1804ВУ3 представляет собой преобразователь, выполненный на ПЗУ емкостью 32 8-разрядных слова. Входными сигналами являются разряды *I0*, ..., *I3* микрокоманды и код условия ветвления TST, указанный в микрокоманде. На восьми управляемых выходах формируется набор управляющих сигналов, соответствующий значению кода микрокоманды (*I0*, ..., *I3*) или условия ветвления (TST).

Схема управления последовательностью микрокоманд К1804ВУ4. 12-разрядная БИС К1804ВУ4 (рис. 10.20) используется для построения блоков микропрограммного управления различных цифровых устройств и предназначена для формирования адресов микрокоманд под воздействием внешних управляющих сигналов. БИС К1804ВУ4 обеспечивает:

адресацию 4096 ячеек микропрограммной памяти;

получение адреса микрокоманды из четырех источников (внутренний регистр адреса/счетчик, счетчик микрокоманд, адресная шина, стек);

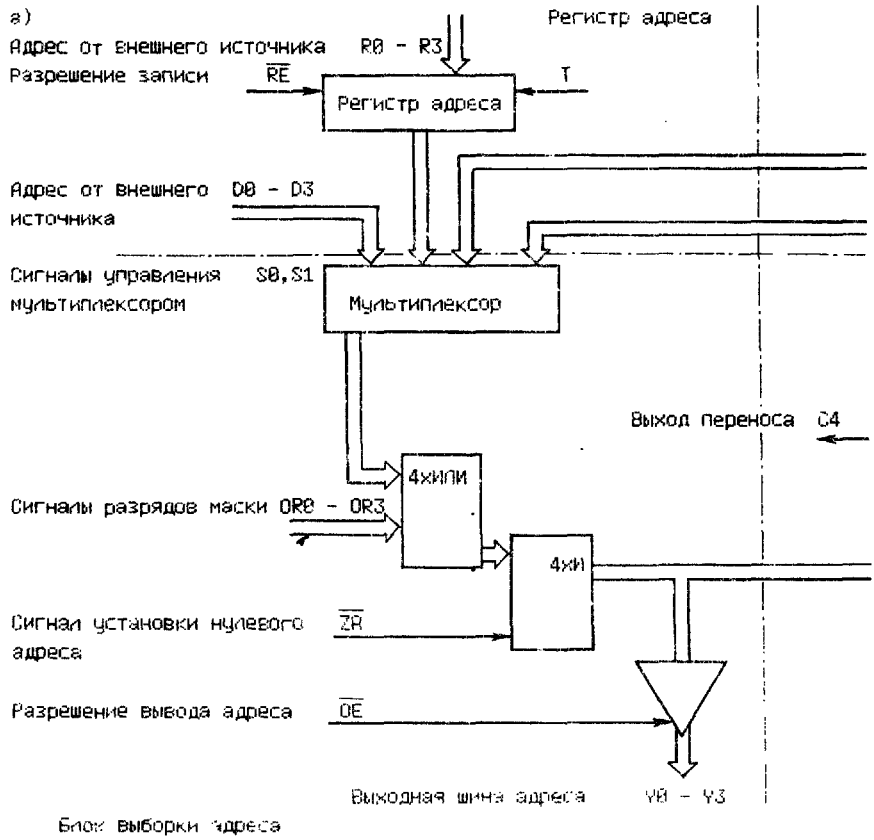


Рис. 10.18 Секции управления адресом
а — структурная схема БИС К1804ВУ1;

реализацию 16 инструкций управления последовательностью выполнения микрокоманд;

управление работой трех внешних устройств (регистров), подключенных к адресной шине.

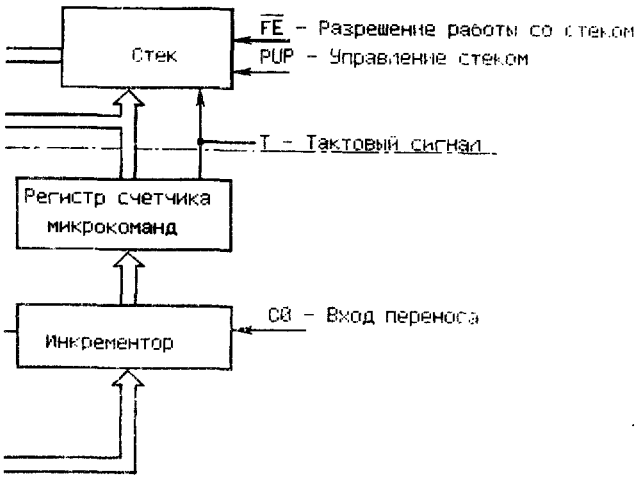
В составе БИС можно выделить следующие шесть основных блоков.

Мультиплексор обеспечивает выбор одного из четырех источников адреса следующей микрокоманды в зависимости от значения разрядов *I0*, ..., *I3* кода микрокоманды и управляющих сигналов $\overline{CC}$ и $\overline{CCE}$.

Регистр адреса/счетчик служит в качестве буфера для записи и хранения адреса или числа циклов, поступающих от внешнего источника по шине *D0*,...,*D11*. Он может также использоваться как счетчик циклов, содержимое которого уменьшается на единицу в каждом такте.

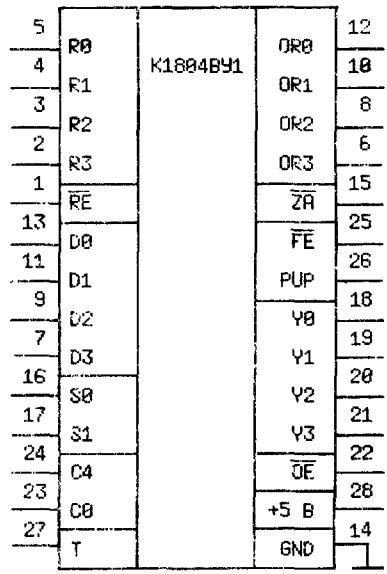
Формирователь признака нуля вырабатывает внутренний управляющий сигнал, когда содержимое регистра адреса/счетчика становится равным нулю.

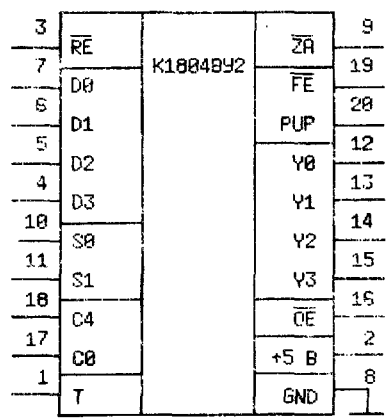
Счетчик микрокоманд, состоящий из регистра счетчика микрокоманд и инкрементора, предназначен для хранения выработанного БИС адреса микрокоманды и (при необходимости) увеличения адреса на единицу.



Счетчик микрокоманд

б)





микрокоманды К1804ВУ1 и К1804ВУ2:
б — нумерация выводов БИС К1804ВУ1 и К1804ВУ2

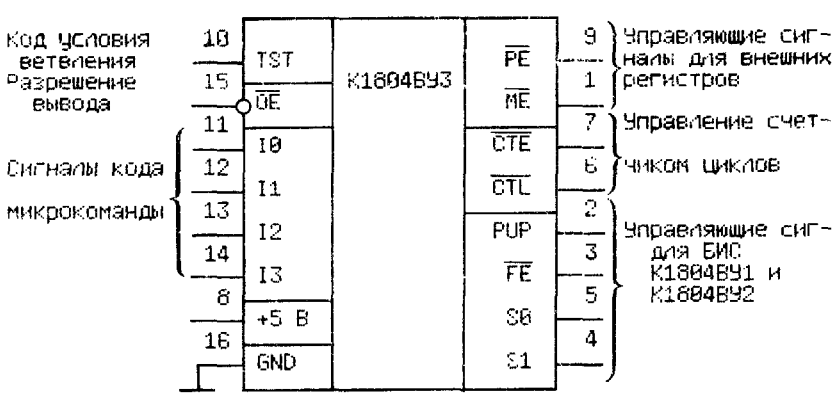


Рис. 10.19. Функциональное назначение и нумерация выводов схемы управления следующим адресом К1804ВУ3

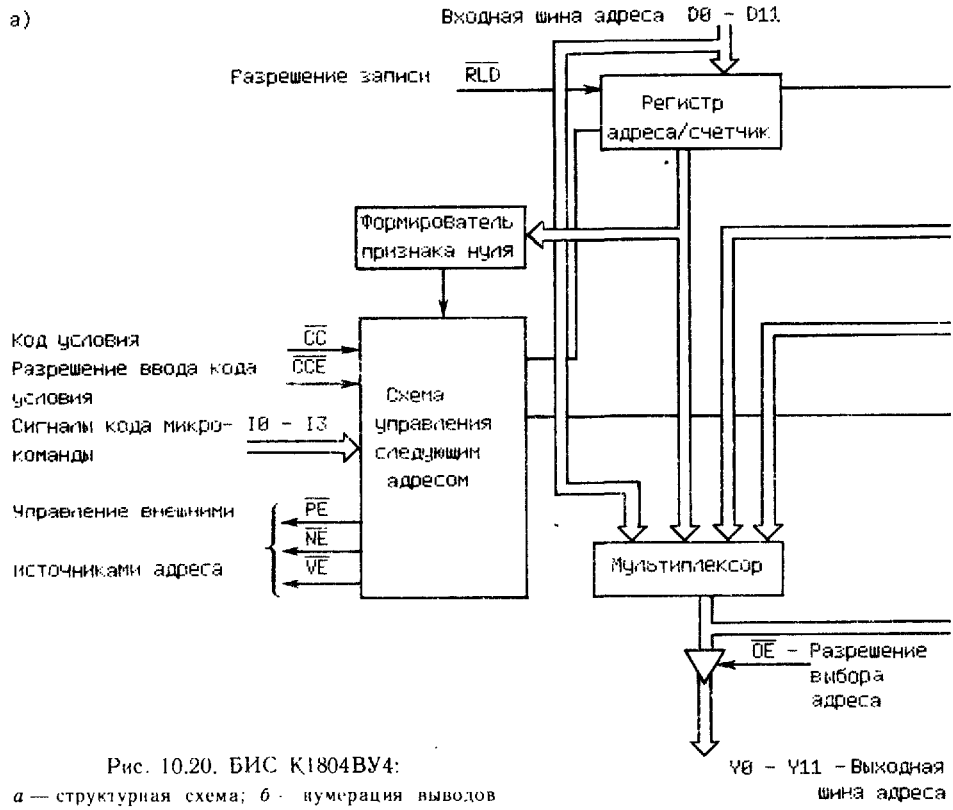


Рис. 10.20. БИС К1804ВУ4:
а — структурная схема; *б* — нумерация выводов

Стек емкостью пять 12-разрядных слов используется для хранения адреса возврата при выполнении подпрограммы и реализации циклов.

Схема управления следующим адресом обеспечивает формирование внутренних управляющих сигналов для всех блоков БИС и трех сигналов для управления внешними источниками адреса, подключенными к адресной шине *D0,...,D11*.

10.6. Пример применения микропроцессорного комплекта БИС КР580

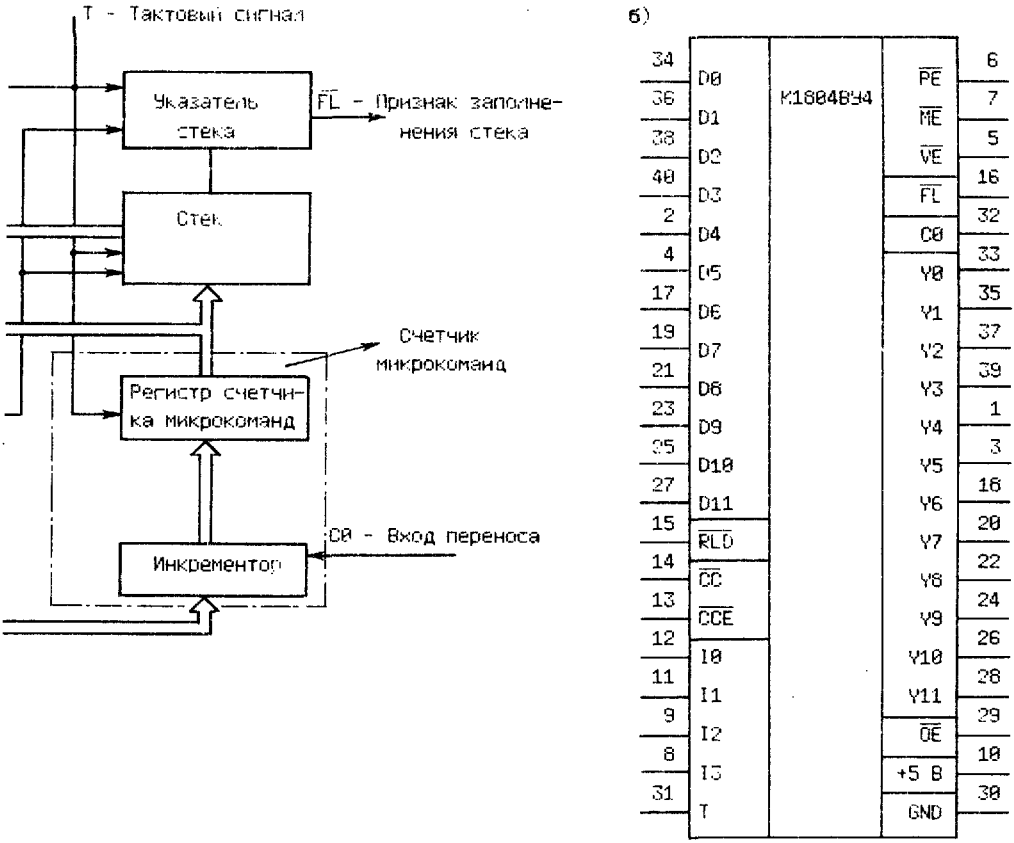
Рассмотрим построение управляющей микроЭВМ для системы функционального моделирования цифровых устройств (см. параграф п. 9.2).

Система функционального моделирования цифровых устройств включает в свой состав:

микроЭВМ, оснащенную комплексом программ моделирования процессов функционирования цифровых устройств различных типов;

пульт управления с набором тумблеров (40 шт.) и кнопок (8 шт.), которые отображают органы управления моделируемого цифрового устройства.

Тумблеры и кнопки (см., например, рис. 3.1) используются для ввода данных в моделирующую программу и для управления процессом моделирования (режимами работы моделируемого устройства). Функциональное назначение сигналов, формируемых на пульте управления, определяется типом моделируемого цифрового устройства;



матричный индикатор, представляющий собой информационное поле с 4096 (64×64) элементами изображения и предназначенный для отображения результатов моделирования.

Результаты моделирования представляются на мнемонической схеме цифрового устройства, наложенной на индикатор, в двоичном коде в виде содержимого (состояний) отдельных компонентов моделируемого устройства — триггеров, регистров, ячеек памяти, шин и т. п.

МикроЭВМ в системе функционального моделирования должна обеспечивать помимо реализации программ моделирования прием информации от пульта управления и выдачу результатов моделирования на матричный индикатор. Следовательно, прежде всего необходимо решить вопрос о форматах данных, которыми микроЭВМ будет обмениваться с пультом управления и индикации.

От тумблеров и кнопок пульта управления может поступать до 48 двоичных сигналов, причем, как уже отмечалось, функциональное назначение отдельных сигналов, а следовательно, и их группировка в отдельные слова информации определяются программой моделирования (типом моделируемого цифрового устройства). Таким образом, микроЭВМ в системе функционального моделирования должна обеспечивать ввод отдельными словами 48 двоичных сигналов от пульта управления. Наиболее типичными для различных цифровых устройств будут слова информации разрядностью 2, 4, 8 или 16 бит.

Проще обстоит дело с форматами данных, выводимых на индикатор. В качестве индикатора в системе функционального моделирования исполь-

зуется функциональное законченное устройство ИМГ-3, оснащенное собственной системой управления и памятью изображения емкостью 4096 бит (запоминающих элементов). Для того чтобы «зажечь» отдельный элемент изображения на индикаторе, достаточно записать в соответствующий запоминающий элемент 1, чтобы «погасить» — записать 0. Всего устройство управления матричного индикатора выполняет четыре команды:

«Запись» — запись 1 в адресуемый запоминающий элемент;

«Стирание» — запись 0 в адресуемый запоминающий элемент;

«Индикация» — отображение содержимого памяти изображения на информационном поле;

«Сброс» — запись 0 во все запоминающие элементы индикаторы.

Формат команды для матричного индикатора приведен на рис. 10.21.

Таким образом, программа моделирования должна формировать для индикатора команды установленного формата, а аппаратные средства системы ввода-вывода микроЭВМ должны обеспечивать их передачу.

Режимы обмена данными с пультом управления и индикатором выбираются из следующих соображений. Устройство управления матричного индикатора обеспечивает прием и выполнение очередной команды не чаще чем через 5 мкс. Следовательно, если время выполнения одной команды микроЭВМ будет не менее 5 мкс, то передачу информации на индикатор можно вести в синхронном режиме обмена без проверки сигнала готовности индикатора.

Прием информации от тумблеров и кнопок пульта управления следует осуществлять как в режиме прерывания программы, так и в асинхронном режиме обмена. Тем самым будут обеспечены любые режимы ввода информации в моделируемое цифровое устройство. Сигнал «Запрос прерывания» в этом случае может формироваться при нажатии одной или нескольких кнопок, а в качестве сигнала «Готовность ВУ» можно использовать состояние любого тумблера пульта управления.

Для размещения программ моделирования в специализированной микроЭВМ целесообразно иметь постоянное ЗУ (ПЗУ) или перепрограммируемое ЗУ (ППЗУ). В этом случае отпадает необходимость в оснащении микроЭВМ внешним устройством для ввода программ. Замена программ моделирования может осуществляться либо заменой ПЗУ (платы или отдельных БИС), либо перезаписью программы в ППЗУ.

Однако кроме постоянной памяти в микроЭВМ необходима память для временного хранения вводимой и выводимой информации, промежуточных результатов вычислений, модифицируемых массивов данных и т. п. Поэтому микроЭВМ следует оснастить и оперативным запоминающим устройством (ОЗУ).

Сведем воедино требования, предъявляемые к управляющей микроЭВМ для системы функционального моделирования цифровых устройств:

Рис. 10.21. Формат команды для матричного индикатора

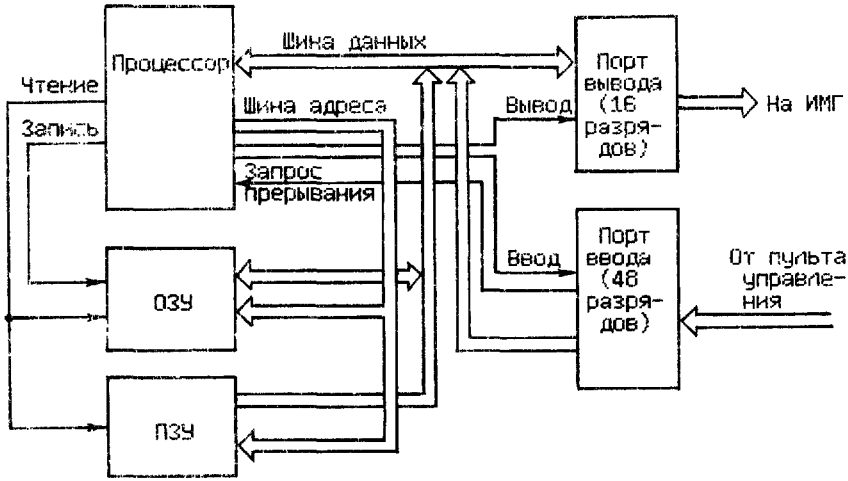


Рис. 10.22. Функциональная схема управляющей микроЭВМ для системы функционального моделирования цифровых устройств

наличие в микроЭВМ как ПЗУ (ППЗУ), так и ОЗУ;

реализация параллельного канала вывода (порта вывода) информации для связи с ИМГ (16 разрядов, синхронный режим обмена);

реализация параллельного канала ввода (порта ввода) информации от пульта управления (всего 48 разрядов, асинхронный режим обмена и обмен в режиме прерывания программы).

Функциональная схема управляющей микроЭВМ, отвечающей заданным требованиям, приведена на рис. 10.22.

Рассмотрим возможность построения управляющей микроЭВМ на основе универсального микропроцессорного комплекта БИС КР580 (см. параграф 10.2).

Микропроцессор КР580ИК80А имеет развитую систему команд, которая обеспечивает реализацию эффективных программ моделирования широкой номенклатуры цифровых устройств (см. приложение Г). 16-разрядная шина адреса микропроцессора позволяет оснастить микроЭВМ памятью (ПЗУ и ОЗУ) емкостью до 64 Кбайт и подключать к нему до 256 8-разрядных портов ввода и до 256 8-разрядных портов вывода.

В качестве портов ввода-вывода могут быть использованы БИС КР580ВВ55 — программируемые параллельные интерфейсы, которые обеспечат требуемые в микроЭВМ режимы обмена с пультом управления и индикации.

Структурная схема управляющей микроЭВМ на основе МПК БИС КР580 приведена на рис. 10.23 *.

Архитектура микропроцессора КР580ИК80А ориентирована на построение микроЭВМ с раздельными шинами адреса и данных (см. параграф 5.2 и рис. 5.2, *б*). При этом адресация ВУ (портов ввода-вывода) может осуществляться по шине адреса либо для адресации ВУ организуется отдельная шина. Для выбора того или иного варианта адресации БУ необходимо определить объем основной памяти (ППЗУ и ОЗУ) микроЭВМ, достаточный для размещения и реализации моделирующих программ.

---

* Рассматриваемая микроЭВМ разработана и предложена в качестве примера А. Е. Платуновым.

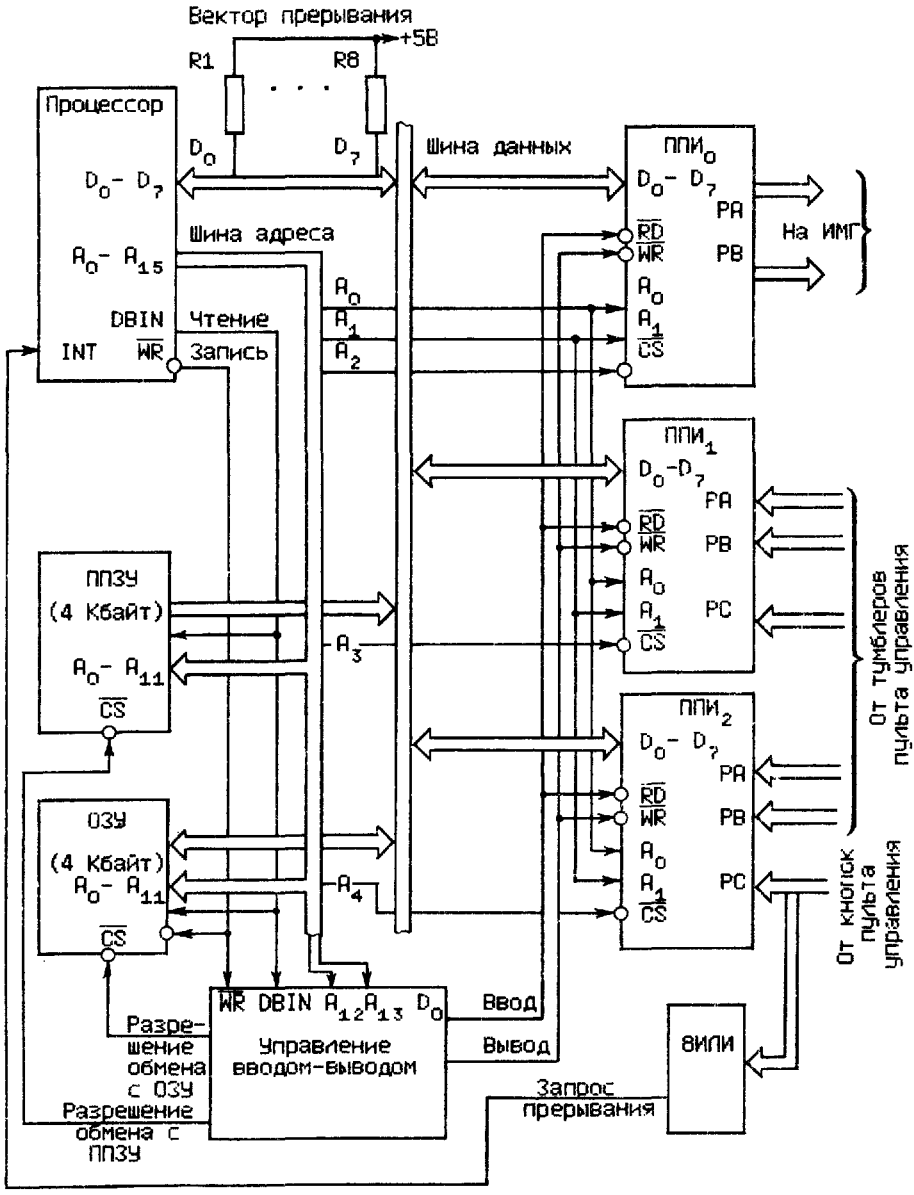


Рис. 10.23. Структурная схема управляющей микроЭВМ

Практическая реализация программных моделей базовой ЭВМ показала, что в управляющей микроЭВМ достаточно иметь ППЗУ и ОЗУ емкостью по 4 Кбайт.

Для обращения к любой 8-разрядной ячейке памяти, имеющей емкость 8 Кбайт, потребуется 12-разрядный адрес. Следовательно, часть линий шины адреса микропроцессора остается свободной и их можно использовать для разделения обращений по вводу-выводу к основной памяти и портам ввода-вывода. В разрабатываемой микроЭВМ для этого используются линии A_{12} и A_{13} шины адреса. По значениям сигналов в этих линиях схема

управления вводом-выводом вырабатывает нижеследующие управляющие сигналы:

«Разрешение обмена с ППЗУ» (выбор кристалла $\overline{CS}$) при $A_{13}=0$, $A_{12}=1$,

«Разрешение обмена с ОЗУ» (выбор кристалла $\overline{CS}$) при $A_{13}=1$, $A_{12}=0$,

«Ввод» или «Вывод» для параллельных каналов ввода-вывода при $A_{13}=1$, $A_{12}=1$ в зависимости от сигналов «Запись» ($\overline{WR}$) и «Чтение» ($DBIN$) микропроцессора.

Параллельные каналы ввода-вывода микроЭВМ реализованы на нескольких БИС программируемого параллельного интерфейса (ППИ) КР580ВВ55 (см. параграф 10.2 и рис. 10.1). Для передачи информации в ИМГ используется $ППИ_0$, в котором задействованы только два 8-разрядных порта вывода PA и PB, а порт PC не используется. Таким образом, с помощью одной БИС КР580ВВ55 организуется передача на ИМГ 16-разрядного управляющего слова.

$ППИ_0$ адресуется по линии A_2 адресной шины микропроцессора. Для приема информации от пульта управления используются $ППИ_1$ и $ППИ_2$, в каждом из которых задействованы все три 8-разрядных порта — PA, PB и PC, что обеспечивает ввод до 48 двоичных сигналов от пульта управления. $ППИ_1$ и $ППИ_2$ адресуются соответственно по линиям A_3 и A_4 шины адреса микропроцессора. Адресация отдельных портов PA, PB и PC в каждом $ППИ$ производится по линиям A_0 и A_1 шины адреса. Таким образом, линии A_0, ..., A_4 шины адреса микропроцессора используются при адресации как ячеек памяти (ППЗУ и ОЗУ), так и портов ввода-вывода. Разделение адресов, относящихся к ППЗУ, ОЗУ или к портам ввода-вывода производится с помощью адресных сигналов A_{12} и A_{13}.

Настройка портов PA и PB $ППИ_0$ на синхронный режим обмена для вывода информации в ИМГ и всех портов $ППИ_1$ и $ППИ_2$ на асинхронный режим обмена для ввода информации от пульта управления и индикации (ПУИ) осуществляется программным путем перед началом обмена с ПУИ.

Ввод информации с пульта управления в режиме прерывания программы реализуется в микроЭВМ следующим образом. Сигналы от всех кнопок пульта подаются в порт PC $ППИ_2$ и одновременно на восьмивходовую схему ИЛИ. При нажатии на любую кнопку на выходе схемы ИЛИ формируется сигнал «Запрос прерывания», который подается на вход INT микропроцессора.

Обрабатывая сигнал «Запрос прерывания», микропроцессор формирует управляющий сигнал «Подтверждение прерывания» и выдает его в линию D_0 шины данных (см. табл. 8.1). По этому сигналу схема управления вводом-выводом блокирует все источники информации в микроЭВМ (ППЗУ, ОЗУ и ППИ) на один цикл обмена. Микропроцессор в этом цикле обмена принимает вектор прерывания — команду RST7 («Повторный запуск»), код которой $(FF)_{16}$ формируется подачей +5 В (логическая 1) на шину данных резисторами $R1$, ..., $R8$. Команда RST7 (см. приложение Г) обеспечивает сохранение в ОЗУ адреса возврата в основную программу и организует переход к подпрограмме обработки прерывания, которая располагается в памяти начиная с адреса $(0038)_{16}$. Ниже приведен фрагмент программы обработки прерывания.

```
BEG:PUSH PSW   ;сохранение содержимого
    PUSH H     ;       регистров
    PUSH D     ;   микропроцессора
    PUSH B     ;     в стеке ОЗУ
```

```
    LDA C(ППИ2);чтение содержимого порта РС ППИ2 в аккумулятор
    LXI B,8    ;установка счетчика сдвигов
M1: DCR C      ;модификация (-1) счетчика сдвигов
    RLC        ;циклический сдвиг влево содержимого аккумулятора
    JNC M2     ;переход на формирование адреса функции, если есть
               ;единица переноса, соответствующая нажатой клавише
    JNZ M1     ;переход на продолжение сдвигов, если нет единицы переноса
    JMP ENI    ;аварийный выход из программы
M2: LXI H,TABF ;формирование в регистрах H,L
    DAD B      ;        адреса строки
    DAD B      ;      таблицы переходов
    MOV E,M    ;формирование в регистрах H,L
    INX H      ;       адреса функции,
    MOV D,M    ;       соответствующей
    XCHG       ;       нажатой  кнопке
    PSHL       ; переход на исполнение функции, соответствующей нажатой
               ;кнопке, запись содержимого регистров H,L в счетчик команд
ENI:POP B      ; восстановление
    POP D      ;  содержимого
    POP H      ;   регистров
    POP PSW    ;микропроцессора
    EI         ;разрешение прерывания
    RET        ;возврат из подпрограммы обработки прерывания
```

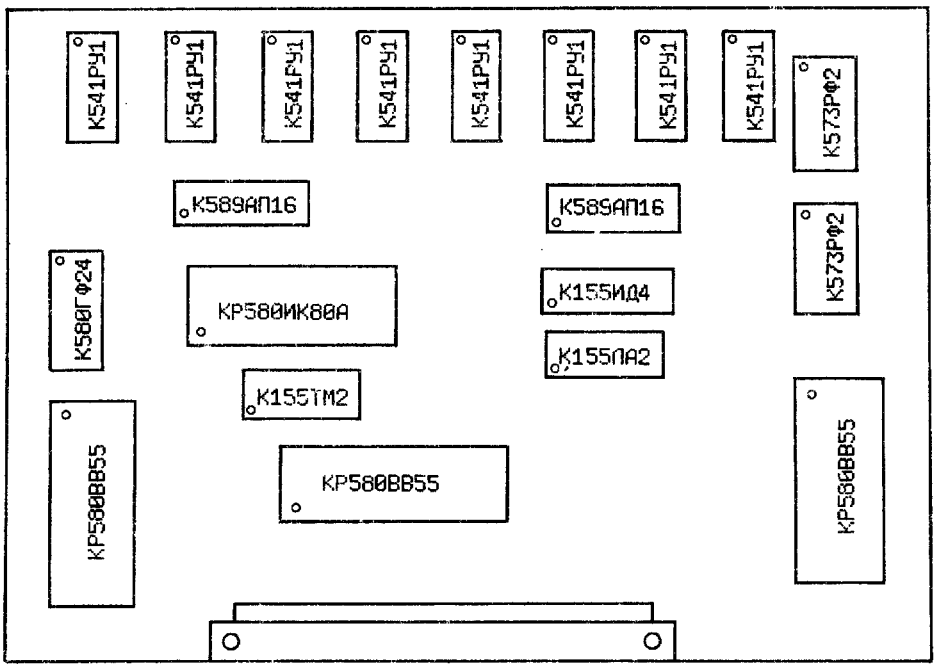


Рис. 10.24. Размещение на печатной плате элементов управляющей микроЭВМ

ППЗУ микроЭВМ реализуется на двух ИС К573РФ2, а ОЗУ — на восьми ИС К541РУ1, для согласования которых с двунаправленной шиной данных микропроцессора необходимо использование шинных формирователей К589АП16. Характеристики указанных ИС приведены в любом справочнике по интегральным микросхемам, например [43].

Всего в состав микроЭВМ входит 21 интегральная схема, размещение которых на печатной плате размерами 180×140 мм показано на рис. 10.24.

В рассмотренном примере отражены только основные вопросы, связанные с функциональной и структурной организацией специализированной микроЭВМ на основе МПК БИС КР580. Это, в первую очередь, реализация системы ввода-вывода микроЭВМ и различных режимов обмена информацией с ВУ. Более полно вопросы применения МПК БИС КР580 освещены в работе [1].

Глава 11

СОВРЕМЕННЫЕ МИКРОЭВМ

11.1. Классификация микроЭВМ

В настоящее время термин микроЭВМ связывается с двумя различными типами средств вычислительной техники:

универсальным блоком обработки данных и выработки управляющих сигналов (серийно выпускаемым в виде одной БИС, одной платы или небольшого прибора, разработанных и созданных на основе микропроцессорных комплектов БИС), предназначенным для встраивания в различные специализированные системы контроля и управления, в бытовую технику и другую аппаратуру в целях расширения ее функциональных возможностей, повышения «интеллектуального» уровня, производительности, надежности, а также улучшения других характеристик и придания новых (ранее отсутствующих) свойств;

очень надежной универсальной ЭВМ малых габаритов (персональной ЭВМ), предназначенной для работы в интерактивном (диалоговом) режиме с индивидуальным пользователем, простой в освоении и использовании (даже теми, кто не знаком с вычислительной техникой), имеющей большую библиотеку программ для различных применений (в науке, промышленности, учрежденческой деятельности, медицине, торговле, журналистике, школе, вузе, быту и т. д.), доступной для приобретения индивидуальным пользователем (бытовой компьютер или бытовая персональная ЭВМ стоимостью 300—1000 руб.) или организацией для использования в профессиональной деятельности своих сотрудников (профессиональная персональная ЭВМ стоимостью 5—30 тыс. руб.).

Таким образом, микроЭВМ — это совершенно новый современный класс технических устройств, предназначенных для обработки разнообразной информации, зачастую не уступающих по своим функциональным возможностям ЭВМ других типов и отличающихся исключительно малыми габаритами, низкой стоимостью и малым потреблением энергии.

Создание и бурное развитие микроЭВМ стали возможны только с появлением и развитием интегральной микросхемотехники (см. гл. 10 и параграф 5.5). Однокристальные микропроцессоры, микропроцессорные комплекты интегральных схем, однокристальные микроЭВМ существенным, если

не сказать революционным, образом расширили сферы применения средств вычислительной техники в различных областях науки, техники, образования и т. д. Стало возможным использование средств микропроцессорной техники (микроЭВМ) практически везде, где требуются компактные, надежные и экономичные устройства управления и обработки информации — в измерительных приборах, фото- и кинокамерах, карбюраторах автомобильных двигателей, лифтах, кассовых аппаратах, пишущих машинках и т. д. и т. п.

Такое разнообразие областей применения и тот факт, что микроЭВМ является универсальным, программно-управляемым устройством, т. е. настраивается на выполнение тех или иных функций с помощью хранимой в ее памяти программы, указывает на то, что микроЭВМ удобнее классифицировать не по возможным областям применения, а по способу использования и конструктивному исполнению (комплектации внешними устройствами). С этими же признаками классификации, в первую очередь с комплектацией внешними устройствами, связаны зачастую и фукциональные возможности микроЭВМ различных типов.

Все микроЭВМ, выпускаемые отечественной промышленностью, можно разделить на три основных класса: встраиваемые, автономные и персональные ЭВМ.

Характерным признаком встраиваемых микроЭВМ является отсутствие, как правило, в ее составе средств связи с пользователем. МикроЭВМ связана только с датчиками, вырабатывающими входные сигналы, и с объектами управления. Программы работы встраиваемых микроЭВМ размещаются в ПЗУ. Конструктивно встраиваемые микроЭВМ выполняют в виде отдельных БИС либо на основе МПК БИС различных серий в виде отдельной платы или набора плат с разъемами для подключения к электронным схемам изделия, в которое они встраиваются.

Автономные микроЭВМ выполняют на основе МПК БИС в виде конструктивно законченного изделия с собственным блоком питания. Такие микроЭВМ оснащают различными внешними устройствами: дисплеями, устройствами печати, накопителями на гибких магнитных дисках и магнитной ленте, устройствами сопряжения с каналами связи и т. д. Автономные микроЭВМ имеют, как правило, развитое программное обеспечение. Зачастую автономные микроЭВМ используют для разработки программного обеспечения встраиваемых микроЭВМ, программно с ними совместимых.

Персональные ЭВМ — это, по сути дела, автономные микроЭВМ, своей конструкцией, набором внешних устройств и составом программного обеспечения ориентированные на работу отдельного пользователя.

Перечисленные здесь классы микроЭВМ подробнее рассмотрены в последующих параграфах главы, а программное обеспечение микроЭВМ — в гл. 12.

11.2. Встраиваемые микроЭВМ

Основное назначение встраиваемых микроЭВМ — замена жесткой логики на программируемую при управлении работой различных устройств, приборов, технологического оборудования и т. д. В практике разработки и применения встраиваемых микроЭВМ, сложившейся к настоящему времени, можно выделить три основных направления.

1. Одноплатные микроЭВМ без средств связи со стандартными внешними устройствами, ориентированные на самостоятельное использование. Такого типа микроЭВМ принято называть микроконтроллерами.

2. Одноплатные микроЭВМ, предназначенные как для самостоятельного использования, так и в составе многоплатных автономных микроЭВМ.

3. Однокристальные микроЭВМ.

К типичным современным **микроконтроллерам** относятся одноплатные микроЭВМ семейства «Электроника С5-41» («Электроника СМС 121»). Микроконтроллеры (МК) этого семейства являются функционально законченными устройствами, имеют программно-управляемые каналы связи с объектами управления, объемы ПЗУ (ППЗУ), достаточные для хранения программ работы МК и исключающие необходимость ввода программ с внешних носителей.

МК «Электроника С5-41» выпускают в виде двух базовых моделей: закрытой и открытой.

Закрытая модель МК «Электроника СМС 12101.1» ориентирована на самостоятельное использование и рассчитана на связь с объектом управления только по программируемым цифровым каналам ввода-вывода.

В **открытой модели** МК «Электроника СМС 12102.1» предусмотрены средства для расширения ее функциональных возможностей за счет подключения дополнительных функциональных модулей через межмодульный параллельный интерфейс (МПИ) и объединения нескольких МК в малую локальную сеть по системному последовательному каналу.

Основой обеих моделей МК является базовый функциональный фрагмент, реализованный на основе БИС серии К1801 и К1809. В состав базового функционального фрагмента МК входят следующие функциональные устройства:

1) однокристальный микропроцессор К1801ВМ1;

2) статическое ОЗУ емкостью 1К 16-разрядных слов, реализованное на БИС К1809РУ1;

3) ПЗУ емкостью 4К 16-разрядных слов, реализованное на БИС К1809РЕ1; ПЗУ используется для хранения резидентного программного обеспечения микроконтроллера;

4) розетка для установки дополнительных БИС памяти: К1809РУ1, К1809РЕ1 или ППЗУ К573РФ3 емкостью 4К 16-разрядных слов;

5) радиальный последовательный интерфейс на основе БИС К1801ВП1-035;

6) магистральный последовательный интерфейс на основе БИС К1809ВВ2;

7) тактовый генератор на основе МС К531ЛН1Т.

Реализация закрытой или открытой модели МК производится за счет переменных функциональных фрагментов (специальных БИС), устанавливаемых в отведенные для них четыре розетки на плате МК. В набор переменных функциональных фрагментов входят: межмодульный параллельный интерфейс, цифровые каналы ввода-вывода (К1809ВВ1), интерфейс магнитофона (К1809ВГ1), интерфейс связи с дисплеем (К1809ВГ3) и др.

Обмен информацией с электронными схемами объекта управления осуществляется в МК через цифровые каналы ввода-вывода в программно-управляемом режиме: асинхронном либо в режиме прерывания программы.

Рассматриваемые МК имеют систему команд микроЭВМ «Электроника-60» (см. приложение В) и являются программно-совместимыми со всеми автономными микроЭВМ типа «Электроника-60» и мини-ЭВМ семейства СМ3,4. Таким образом, любая из указанных микро- и мини-ЭВМ может использоваться для разработки программного обеспечения микроконтроллеров семейства «Электроника С5-41».

МК семейства «Электроника С5-41» реализуются на платах размерами 273×173 мм, на которых размещаются как элементы базового функционального фрагмента, так и розетки для подключения БИС из набора пере-

менных функциональных фрагментов. Собственного блока питания микроконтроллеры не имеют.

Более подробно с микроконтроллерами семейства «Электроника С5-41», наборами функциональных фрагментов и модулей можно ознакомиться в работе [31].

Отечественная промышленность выпускает целый ряд одноплатных микроЭВМ, которые могут использоваться как самостоятельно в качестве встраиваемых микроЭВМ, так и в составе многоплатных автономных микроЭВМ. Типичными и наиболее распространенными моделями встраиваемых одноплатных микроЭВМ являются «Электроника НЦ-80-01Д» и ряд микроЭВМ «Электроника МС1201».

Все эти микроЭВМ построены на основе однокристального микропроцессора К1801ВМ1 (модель «Электроника МС1201.02» — на основе микропроцессора К1801ВМ2) и других БИС МПК К1801 и реализуют систему команд микроЭВМ «Электроника-60». Тем самым решается проблема автоматизации программирования для встраиваемых микроЭВМ, не имеющих средств связи с пользователем. Программы, настраивающие встраиваемую микроЭВМ на реализацию заданных функций, размещаются в ПЗУ емкостью от 8 до 16 Кбайт.

Рассматриваемые микроЭВМ обмениваются информацией с объектами управления по параллельному (ИРПР) либо последовательному (ИРПС) каналу связи в программно-управляемом режиме обмена. Вместе с тем во всех этих микроЭВМ реализован стандартный системный интерфейс микроЭВМ «Электроника-60» (Q-bus), что обеспечивает возможность их использования в качестве центральных процессоров в многоплатных автономных микроЭВМ (см. параграф 11.3).

МикроЭВМ «Электроника НЦ-80-01Д» и микроЭВМ ряда «Электроника МС1201» реализованы на стандартной плате микроЭВМ «Электроника-60» размерами 292×256 мм, что существенно облегчает использование указанных микроЭВМ в различных устройствах программного управления совместно с функциональными модулями микроЭВМ «Электроника-60» [20, 32].

Наиболее дешевыми и массовыми средствами программного управления различными приборами и устройствами являются однокристальные микроЭВМ. Отечественная промышленность освоила и выпускает целый ряд однокристальных микроЭВМ различной разрядности от 4 до 16 бит. Все эти микроЭВМ содержат на одном кристалле функционально полные наборы устройств, типичные для микроЭВМ: процессор, память (ОЗУ и ПЗУ), контроллеры ввода-вывода. Это делает их законченными устройствами управления, имеющими предельно малые габариты, высокую надежность и экономичность.

Рассмотрим некоторые модели однокристальных микроЭВМ, обращая в первую очередь внимание на их потребительские характеристики: емкость памяти, количество и разрядность каналов ввода-вывода информации, систему команд, средства для подготовки программ.

Однокристальные 4-разрядные микроЭВМ серии К1814 имеют в своем составе ПЗУ емкостью 1 Кбайт и ОЗУ емкостью 64 4-разрядных слова. Для вывода данных и управляющих сигналов используются два порта вывода: на 13 выводов и 8 выводов. Ввод данных производится через 4-разрядный порт ввода.

Система команд микроЭВМ серии К1814 состоит из 43 команд, обеспечивающих выполнение арифметических и логических операций, операций работы с битами, операций пересылки и ввода-вывода информации. Подготовка программ для микроЭВМ серии К1814 производится на специаль-

ном отладочном устройстве, имеющем в своей основе микроЭВМ этой же серии. Отладочное устройство может работать как автономно, так и в составе отладочного комплекса на основе микроЭВМ «Электроника-60». Подробное описание микроЭВМ серии К1814 приведено в работе [30].

Однокристальные 4-разрядные микроЭВМ серии К1820 выпускаются в двух модификациях:

КР1820ВЕ1 — микроЭВМ с внешним ПЗУ, используется для отладки программного обеспечения и в изделиях с малой тиражностью;

КР1820ВЕ2 — микроЭВМ с внутренним ПЗУ емкостью 1 Кбайт.

В микроЭВМ серии К1820 реализованы пять каналов для обмена информацией в программно-управляемом режиме (асинхронном и в режиме прерывания программы):

8-разрядный двунаправленный порт ввода-вывода;

4-разрядный двунаправленный порт ввода-вывода;

4-разрядный порт вывода;

4-разрядный порт ввода;

последовательный выход.

Система команд микроЭВМ серии К1820 состоит из 49 команд, включающих арифметические команды, передачи управления, обращения к регистрам и ввода-вывода. Полностью система команд, структурные схемы и примеры применения микроЭВМ серии К1820 приведены в работе [29].

Однокристальные 8-разрядные микроЭВМ серии К1816 выпускаются в трех модификациях:

КМ1816ВЕ39 — микроЭВМ с внешним ПЗУ и ОЗУ емкостью 128 байт;

КМ1816ВЕ48 — микроЭВМ с ПЗУ емкостью 1 Кбайт и ОЗУ емкостью 64 байт;

КМ1816ВЕ49 — микроЭВМ с ПЗУ емкостью 2 Кбайт и ОЗУ емкостью 128 байт.

В каждой микроЭВМ этой серии помимо памяти реализованы: центральный процессор, многоканальный интерфейс ввода-вывода, векторная система прерывания, 8-разрядный таймер-счетчик, устройство синхронизации. Предусмотрена возможность расширения памяти программ (ПЗУ), памяти данных (ОЗУ) и увеличения количества линий ввода-вывода за счет подключения внешних БИС ПЗУ (ОЗУ) и интерфейсных БИС серии КР580.

В микроЭВМ серии К1816 реализованы 27 линий ввода-вывода, 24 из которых объединены в три 8-разрядных порта ввода-вывода. Обмен информацией с объектами управления может осуществляться как в асинхронном режиме, так и в режиме прерывания программы.

МикроЭВМ серии К1816 имеют развитую систему команд, состоящую из 96 команд, которые делятся на три группы:

команды передачи данных, включающие и команды ввода-вывода;

команды преобразования данных, включающие арифметические и логические команды;

команды передачи управления.

Полностью система команд, структурные схемы и режимы работы микроЭВМ серии К1816 приведены в работе [15].

Однокристальная 16-разрядная микроЭВМ «Электроника-С5-31» (К586ВЕ1) является однокристальной реализацией распространенной серии микроЭВМ «Электроника С5» [27], полностью программно с ними совместимой, но с ограниченными возможностями по объему хранимых программ и количеству каналов ввода-вывода. В микроЭВМ «Электроника С5-31» реализованы память программ (ПЗУ) емкостью 1К 16-разрядных слов и память данных (ОЗУ) емкостью 128 16-разрядных слов. Имеются возможности для расширения памяти до 32К 16-разрядных слов за счет подклю-

чения внешних БИС памяти. Помимо этого в микроЭВМ реализована резидентная память для хранения служебных программ: ПЗУ — 2 Кбайт и ОЗУ — 256 байт.

Обмен информацией с объектом управления микроЭВМ осуществляет по пяти каналам ввода-вывода, один из которых последовательный, два 8-разрядных параллельных и два 4-разрядных параллельных. Обмен может производиться как в асинхронном режиме, так и в режиме прерывания программы.

Программная совместимость однокристальной микроЭВМ «Электроника С5-31» с семейством микроЭВМ «Электроника С5» позволяет использовать для разработки ее программного обеспечения существующие системы автоматизации программирования на ЕС ЭВМ и БЭСМ.

Завершая разговор о встраиваемых микроЭВМ, следует отметить, что приведенный здесь краткий обзор, конечно же, не охватывает все многообразие существующих моделей встраиваемых микроЭВМ. Это связано с тем, что широкое внедрение в практику различных МПК БИС резко стимулировало и разработку значительного количества разнообразных моделей встраиваемых микроЭВМ, ориентированных на конкретные области применения. Подтверждением этому может служить постоянно увеличивающееся количество публикаций на эту тему.

11.3. Автономные микроЭВМ

В настоящее время автономные микроЭВМ являются, без преувеличения, наиболее массовым типом ЭВМ, используемым в самых различных отраслях народного хозяйства. Отечественная промышленность выпускает серийно целый ряд моделей автономных микроЭВМ, различающихся своими характеристиками, комплектацией внешними устройствами, конструктивным исполнением, составом программного обеспечения и т. д. Тем не менее можно выделить два основных подхода к разработке автономных микроЭВМ, сложившихся в отечественной практике.

1. Создание автономных микроЭВМ, повторяющих своей архитектурой (структурой, системой команд и типом системного интерфейса) распространенные мини-ЭВМ серии СМ3, 4. К этому типу автономных микроЭВМ относится весь ряд микроЭВМ типа «Электроника-60».

2. Создание автономных микроЭВМ оригинальной архитектуры, базирующихся в первую очередь на микропроцессорные комплекты БИС серий КР580 и К1810. Типичными представителями этого типа автономных микроЭВМ являются СМ-1800, «Электроника К1-10», «Электроника К1-20», «Электроника К1-30» и др.

Рассмотрим подробнее оба типа автономных микроЭВМ.

МикроЭВМ «Электроника-60» является первой моделью в целом ряде программно-совместимых микроЭВМ. К этому ряду относятся как микроконтроллеры «Электроника С5-41», одноплатные микроЭВМ «Электроника НЦ-80-01Д» и «Электроника МС1201», так и многоплатные модульные автономные микроЭВМ «Электроника-60», «Электроника МС1211», «Электроника МС1212», диалоговые вычислительные комплексы «Электроника НЦ-80-20». Все микроЭВМ этого ряда реализуют единую систему команд (см. приложение В), а более развитые модели ряда — и дополнительные команды расширенной арифметики и арифметических операций для чисел с плавающей запятой. Таким образом обеспечивается программная совместимость всего ряда микроЭВМ от младших моделей к старшим.

Все микроЭВМ рассматриваемого ряда используют единый системный интерфейс Q-bus (см. параграф 8.1), что позволило обеспечить и аппарат-

ную совместимость на уровне отдельных модулей (контроллеров ВУ, модулей памяти и т. п.).

Рассматриваемый ряд микроЭВМ является, по сути дела, повторением ряда мини-ЭВМ серии СМ3,4 на современной элементной базе — специализированных микропроцессорных комплектах БИС серий К581, К1801, К1809, К1811. Полная программная совместимость рядов мини-ЭВМ СМ3,4 и микроЭВМ типа «Электроника-60» позволила перенести практически весь накопленный для мини-ЭВМ программный продукт в среду микроЭВМ. Этот факт, а также разнообразие комплектаций микроЭВМ типа «Электроника-60» от встраиваемых одноплатных моделей до вычислительных комплексов с широким набором внешних устройств, низкая стоимость отдельных моделей обеспечили широкое распространение данного ряда микроЭВМ в различных областях науки и техники.

МикроЭВМ «Электроника-60» представляет собой сугубо модульную систему, в которой все модули (функциональные блоки) — интерфейсы, контроллеры ВУ, блоки памяти и т. п.— объединяются системным интерфейсом с мультиплексированными шинами «Адрес/Данные» (см. параграф 5.2). Каждый функциональный блок микроЭВМ «Электроника-60» является конструктивно законченным изделием в виде печатной платы одного из двух типоразмеров: 252×296 или 252×148 мм. Конкретный набор функциональных блоков микроЭВМ определяется пользователем и зависит от используемых в составе микроЭВМ внешних устройств и требуемого объема памяти.

Все блоки микроЭВМ «Электроника-60» помещают в стандартный каркас, предназначенный для установки либо четырех плат размерами 252×296 мм, либо восьми плат размерами 252×148 мм или любого их сочетания. Через специальный разъем к каркасу подключают блок питания. При необходимости в составе микроЭВМ можно использовать два и более соединенных между собой каркасов.

Для микроЭВМ ряда «Электроника-60» выпускают несколько типов одноплатных центральных процессоров — М2, М3, М6, различающихся быстродействием и перечнем выполняемых команд. В качестве центрального процессора можно использовать и одноплатные микроЭВМ «Электроника НЦ-80-20Д» и «Электроника МС1201».

В набор стандартных функциональных блоков для микроЭВМ «Электроника-60» входят [20]:

параллельные интерфейсы, обеспечивающие ввод-вывод информации байтами или 16-разрядными словами в программно-управляемом режиме (платы И2, И7, И9 и др.);

последовательный интерфейс (устройство последовательного обмена — УПО), обеспечивающий обмен информацией в дуплексном режиме со скоростью до 9200 бит/с;

платы памяти — динамические ОЗУ емкостью 8 Кбайт (П2) и 32 Кбайт (П3);

платы перепрограммируемой памяти с электрической перезаписью и ультрафиолетовым стиранием типа УЗПП емкостью от 4 до 16 Кбайт;

аналогоцифровые и цифроаналоговые преобразователи;

интерфейсные платы для подключения стандартных внешних устройств, электрических пишущих машинок, дисплеев, накопителей на гибких магнитных дисках, накопителей на магнитной ленте, ленточных перфоратора и считывателя и др.

Рассмотренные здесь возможности комплектации микроЭВМ типа «Электроника-60» внешними устройствами, ее система команд и используемый в ней системный интерфейс указывают на то, что это современная универсальная микроЭВМ с широким диапазоном применения.

Таблица 11.1

Сравнительные характеристики некоторых моделей микроЭВМ семейства «Электроника-60»

Параметр	«Электроника-60М»	«Электроника МС1211.02»	«Электроника МС1212»
Серия МПК БИС	К581	К1811	
Разрядность		16	
Число команд	91	138	
Объем адресуемой памяти, Кбайт	64	256	4000
Быстродействие при выполнении команд типа «Сложение» при регистровом методе адресации, тыс. команд/с	250	500	580
Число уровней прерывания	1	4	
Контроль по четности	Нет	Есть	
Стартовая самодиагностика	»	»	
Резервирование питания	»	»	
Габаритные размеры, мм	388×100×520	482,6×132×626,2	

МикроЭВМ типа «Электроника-60» постоянно совершенствуют за счет использования современной элементной базы, что приводит к существенному улучшению их функциональных возможностей, как это хорошо видно по приведенным в табл. 11.1 данным.

Более подробно с различными вопросами структурной и функциональной организации микроЭВМ типа «Электроника-60», принципами их работы можно ознакомиться в работах [3, 37, 41].

МикроЭВМ СМ-1800 — 8-разрядная универсальная ЭВМ, построенная на базе МПК БИС серии КР580. СМ-1800 — модульная система, состоящая из набора модулей различного назначения, связанных между собой единым системным интерфейсом И-41.

И-41 — системный интерфейс с изолированными шинами (см. параграф 5.2) — включает 20-разрядную шину адреса и 16-разрядную шину данных. Интерфейс И-41 обеспечивает адресацию до 1 Мбайт памяти и до 8К устройств портов ввода-вывода, позволяет подключать внешние устройства, работающие как в режиме прямого доступа к памяти, так и в программно-управляемом режиме. Интерфейс И-41 допускает одновременное подключение до восьми активных устройств (процессоров), что обеспечивает возможность создания на базе СМ-1800 мультипроцессорных комплексов высокой производительности.

Для микроЭВМ СМ-1800 разработаны более 50 различных модулей. Конструктивно каждый модуль выполняют в виде одного или двух блоков элементов (печатных плат) размерами 233,35×220 мм. Блоки элементов устанавливают в монтажный блок, рассчитанный на размещение 10 блоков элементов. Два монтажных блока совместно с блоком питания объединяются в автономный комплектный блок, являющийся конструктивной основой микроЭВМ СМ-1800 любой комплектации.

Конкретный набор модулей, используемых в микроЭВМ, определяется пользователем и зависит от функций, возлагаемых на микроЭВМ. Все модули, входящие в состав микроЭВМ СМ-1800, можно разделить на несколько групп: процессоры, память, устройства связи с объектом, межмашинного обмена, отображения информации и системные модули и устройства.

Для микроЭВМ СМ-1800 разработаны три типа модулей процессоров. Модуль центрального процессора (МЦП) обеспечивает обработку информации и управление интерфейсом И-41. Система команд модуля определяется микропроцессором КР580ИК80А. В МЦП реализована локальная

память: ПЗУ — 2 Кбайт и ОЗУ — 1 Кбайт. Модуль обеспечивает адресацию до 128 Кбайт памяти и до 512 портов ввода-вывода. Время выполнения команд в модуле от 2 до 8,5 мкс. Конструктивно модуль выполнен в виде двух блоков элементов.

Модуль центрального процессора с повышенным объемом памяти (МЦП1) имеет локальную память (ПЗУ и ОЗУ) до 8 Кбайт и обеспечивает адресацию до 1 Мбайт памяти.

Модуль центрального процессора повышенной производительности (МЦП16) построен на базе 16-разрядного микропроцессора КМ1810ВМ86. Быстродействие модуля в три раза выше, чем у МЦП. Модуль обеспечивает адресацию до 2 Мбайт памяти и до 64К портов ввода-вывода. Система команд модуля определяется микропроцессором КМ1810ВМ86.

Для микроЭВМ СМ-1800 разработаны следующие модули памяти:

модули оперативные запоминающие МОЗ и МОЗ1 емкостью 32 и 64 Кбайт соответственно;

модули постоянные запоминающие МПЗ и МПЗ1 емкостью 4 и 8 Кбайт соответственно;

модуль полупостоянный запоминающий с ультрафиолетовым стиранием МППЗ емкостью 32 Кбайт.

Группа модулей устройств связи с объектом является наиболее представительной и включает:

модули ввода (МВВА и МВвАС) и вывода (МВА) аналоговых сигналов;

модули коммутации аналоговых сигналов (МКАС);

модуль нормализации аналоговых сигналов (МНАС);

модуль компараторов уровня (МКУ);

модули ввода (МВвД) и вывода (МВД и МВДС) дискретных сигналов;

модуль управления цепями переменного тока повышенной мощности (МЦУП) и др.

Даже простой перечень модулей устройств связи с объектами показывает широкие функциональные возможности микроЭВМ СМ-1800 при использовании ее в различных системах управления.

Модули межмашинного обмена, входящие в состав СМ-1800, служат как для связи самих микроЭВМ, так и для связи с ЭВМ типа СМ3,4 и ЕС ЭВМ. В эту группу входят модули связи с параллельными и последовательными интерфейсами (МИРПР и МИРПС); модемом (МСМ), удаленным терминалом (МСД), устройство сопряжения с интерфейсом типа «Общая шина» и др.

Целый ряд модулей микроЭВМ СМ-1800 обеспечивает вывод на телевизионный экран символьной (МВСТ) и графической (МВГТ) информации, подключение алфавитно-цифровых печатающих устройств и перфоленточных устройств ввода-вывода (МИРПР). В качестве внешней памяти в микроЭВМ СМ-1800 используются накопители на гибких магнитных дисках типа PL×45D5.

Более подробную информацию о микроЭВМ СМ-1800, характеристиках ее модулей, программном обеспечении можно получить в работах [26, 35].

11.4. Персональные микроЭВМ

Последние годы развития вычислительной техники ознаменовались появлением нового класса микроЭВМ — персональных ЭВМ, что обусловлено двумя основными факторами: революционным развитием микропроцессорной техники и резко возросшими потребностями в вычислительной технике в различных областях профессиональной деятельности, где ранее ЭВМ не применяли.

Именно эти два фактора и определяют характерные особенности персональных ЭВМ. С одной стороны, все блоки персональных ЭВМ строятся, как правило, с использованием МПК БИС и, как следствие, персональные ЭВМ компактны, обладают высокими функциональными возможностями и потребительскими характеристиками. С другой стороны, персональные ЭВМ просты в эксплуатации и обслуживании, не требуют от пользователя специальной подготовки в области вычислительной техники.

Рассмотрим типичные модели персональных ЭВМ различного назначения и их комплектацию. (Особенности программного обеспечения персональных ЭВМ приведены в параграфе 12.4.)

К настоящему времени разработаны и выпускаются отечественной промышленностью целый ряд моделей персональных ЭВМ от простейших бытовых компьютеров до профессиональных персональных ЭВМ. Вместе с тем можно выделить две базовые конфигурации, которые положены в основу всех моделей персональных ЭВМ.

Простейшая базовая конфигурация персональных ЭВМ включает процессор и клавиатуру, которые нередко объединяются в один блок. При этом в состав процессора входят средства для подключения дисплея и устройства внешней памяти. Простейшая базовая конфигурация положена в основу **бытовой персональной ЭВМ «Электроника БК 0010».** Процессор и клавиатура ЭВМ «Электроника БК 0010» реализованы в едином блоке. В этом же блоке размещается память — ОЗУ емкостью 32 Кбайт. В качестве дисплея используется бытовой цветной телевизор, на экран которого выводятся либо символы, либо графическая информация (256×256 точек). В персональной ЭВМ «Электроника БК 0010» реализованы средства для подключения бытового кассетного магнитофона, который образует ее внешнюю память. Аналогичную конфигурацию, в том числе и комплектацию внешними устройствами, имеет и школьная персональная ЭВМ «Электроника (Тимур)».

Расширенная базовая конфигурация персональных ЭВМ включает процессор, клавиатуру, дисплей и в некоторых случаях накопители на гибких магнитных дисках и (или) жестких дисках типа «Винчестер». Непосредственно в блоке процессора расширенной базовой конфигурации реализуются оперативная память емкостью до 256 Кбайт и постоянная память емкостью до 64 Кбайт, предназначенная для хранения служебных программ (например, интерпретатор языка БЕЙСИК, программы ввода-вывода и др.). Блок процессора имеет также несколько разъемов (обычно не более восьми) для подключения дополнительных модулей, расширяющих функциональные возможности персональной ЭВМ. В число таких дополнительных модулей могут входить блоки памяти (ОЗУ и ПЗУ), стандартные коммуникационные интерфейсы, сетевые адаптеры для подключения персональных ЭВМ в локальные вычислительные сети и т. п.

Все персональные ЭВМ, построенные на основе расширенной базовой конфигурации, имеют в своем составе устройства печати для документирования результатов вычислений, текстов программ, текстовых и табличных документов, а зачастую — и для получения твердых копий черно-белых или цветных изображений с экрана дисплея.

Устройства, входящие в состав персональных ЭВМ, отличаются компактностью, высокой степенью надежности, удобны в эксплуатации и просты в обслуживании. Все это вместе взятое плюс хорошо развитое «дружественное» программное обеспечение создает отличные предпосылки для применения персональных ЭВМ на основе расширенной базовой конфигурации в самых различных областях профессиональной деятельности. Такого типа персональные ЭВМ получили название профессиональных персональных ЭВМ (ПП ЭВМ).

Таблица 11.2

Характеристики некоторых типов персональных ЭВМ

ЭВМ	Тип процессора	Тип дисплея	Объем адресуемой памяти	Языки программирования
«Агат 11»	КМ1810ВМ86	Символьный; графический 640×200 точек	До 4 Мбайт	БЕЙСИК, ФОРТРАН, ПАСКАЛЬ, С
«Искра 1130»			256 Кбайт	МАКРОАССЕМБЛЕР, БЕЙСИК
«Искра 1030»			До 1 Мбайт	То же
ЕС 1840			До 1 Мбайт	БЕЙСИК, ФОРТРАН, ПАСКАЛЬ, С
«Нейрон И9.66»			256 Кбайт	То же
ДВК-3М	К1801ВМ2	Символьный; графический 440×284 точки	До 4 Мбайт	АССЕМБЛЕР, БЕЙСИК, ФОРТРАН, ПАСКАЛЬ
«Электроника-85»	К1811	Символьный; графический 960×240 точек	До 4 Мбайт	БЕЙСИК, ФОРТРАН, ПАСКАЛЬ, РАПИРА, МОДУЛА 2, С, СМОЛТОК
«Электроника БК-0010»	К1801ВМ1	Бытовой телевизор 25 строк×64 символа, 512×256 точек	64 Кбайт	БЕЙСИК, ФОКАЛ

В табл. 11.2 приведены некоторые характеристики нескольких типов отечественных профессиональных персональных ЭВМ.

Для иллюстрации функциональных возможностей ПП ЭВМ рассмотрим подробнее две модели персональных ЭВМ этого типа: широко распространенную в настоящее время ПП ЭВМ «Искра 226» и ПП ЭВМ ЕС 1840, первую из единой серии персональных ЭВМ, выпуск которых будет осуществляться во все увеличивающихся масштабах.

Профессиональная персональная ЭВМ «Искра 226» выпускается к настоящему времени в семи различных исполнениях (комплектациях), которые ориентированы на следующие области применения [34]:

автоматизация плановых расчетов и использование в качестве интеллектуальных терминалов СМ и ЕС ЭВМ — исполнения 1 и 2;

автоматизация административно-управленческих работ и решение планово-экономических задач — исполнение 3;

автоматизация научно-исследовательских и инженерных расчетов — исполнение 4;

автоматизация рабочих мест исследователей-экспериментаторов — исполнения 6 и 7;

автоматизация подготовки, контроля и редактирования управляющих программ для станков с числовым программным управлением и автоматизация проектирования технологических процессов — АРМ ТП.

Основу ПП ЭВМ «Искра 226» составляет процессор, размещенный в блоке настольного исполнения совместно с символьно-графическим дисплеем, логическими узлами, ОЗУ и ПЗУ. В задней стенке блока процессора встроен расширитель на семь мест для установки интерфейсных функциональных блоков (контроллеров), обеспечивающих как подключение внеш-

них устройств, так и реализацию стандартных каналов связи различного назначения. В комплект интерфейсных блоков ПП ЭВМ «Искра 226» входят контроллеры: накопителей на гибких и жестких магнитных дисках, накопителя на магнитной ленте, печатающего устройства и клавиатуры, графопостроителя, ленточных перфоратора и считывателя, параллельных и последовательных интерфейсов. Имеются интерфейсные блоки цифроаналогового и аналогоцифрового преобразователей. Набор интерфейсных блоков и внешних устройств задается исполнением ПП ЭВМ «Искра 226».

ПП ЭВМ «Искра 226» оснащена оперативной памятью емкостью 128 Кбайт. Символьно-графический дисплей позволяет выводить на экран 24 строки по 80 символов в каждой и изображение размерами 512×256 точек.

Программирование для ПП ЭВМ «Искра 226» осуществляется либо на ассемблере, либо на широко распространенном алгоритмическом языке программирования БЕЙСИК.

ПП ЭВМ ЕС 1840 открывает в единой системе электронных вычислительных машин направление персональных профессиональных ЭВМ. ПП ЭВМ ЕС 1840 представляет собой ЭВМ настольного типа, предназначенную для работы как в автономном режиме, так и в локальных и глобальных вычислительных сетях. Ее можно использовать в качестве интеллектуального термина ЕС ЭВМ.

В состав ПП ЭВМ ЕС 1840 входят дисплей, клавиатура, накопитель на гибких магнитных дисках, печатающее устройство. ЕС 1840 имеет модульное конструктивное исполнение и унифицированный интерфейс с возможностью расширения. Тем самым обеспечивается возможность применения дополнительных модулей профессиональной ориентации и дополнительных внешних устройств, замены одного типа внешних устройств на другие, более перспективные (например, применение цветного графического дисплея, печатающего устройства типа «Ромашка» и т. п.).

ПП ЭВМ ЕС 1840 оснащается оперативной памятью емкостью от 256 Кбайт до 1 Мбайт и ПЗУ емкостью 16 — 64 Кбайт. Символьно-графический дисплей позволяет выводить 25 строк по 80 символов и изображение размерами 640×200 точек.

Программное обеспечение ПП ЭВМ ЕС 1840 включает развитую операционную систему М86, обеспечивающую эффективную работу с носителями информации, настройку на параметры конкретной модификации ЭВМ, а также систему программирования БЕЙСИК М86, в основу которой положен интерпретатор языка БЕЙСИК, дополненный средствами графики. В программное обеспечение ПП ЭВМ ЕС 1840 входит пакет прикладных программ «АБАК», предоставляющий широкие возможности обработки информации, заданной в табличном виде: бухгалтерские и инженерные расчеты, оперативное планирование и др.

Более подробно ознакомиться с различными типами персональных ЭВМ, перспективами их развития и применения можно по работам [4, 39, 40].

11.5. Локальные вычислительные сети микроЭВМ

Интенсивное развитие микропроцессорной техники обеспечило возможность создания и внедрения в различные отрасли народного хозяйства высокопроизводительных микроЭВМ различных классов: встраиваемых, автономных, персональных. Однако круг задач, подлежащих автоматизации на основе средств микропроцессорной техники, необычайно широк и включает такие задачи, реализация которых возможна только при совместном использовании достаточно большого количества микроЭВМ, объединенных в единые системы. К числу таких задач относятся:

автоматизация административной и управленческой деятельности;

гибкие переналаживаемые автоматизированные производства (ГАП);

системы автоматизированного проектирования, в том числе и интегрированные с ГАП;

системы автоматизации научных исследований;

информационно-справочные системы;

автоматизированные обучающие системы.

Одним из наиболее эффективных путей реализации перечисленных выше и многих других задач является объединение микроЭВМ различных классов в локальные вычислительные сети (ЛВС). В ЛВС объединяют микроЭВМ, персональные ЭВМ и другие средства вычислительной техники, расположенные в помещении, здании или группе зданий на расстоянии 1 2 км друг от друга.

Объединение микроЭВМ (персональных ЭВМ) в ЛВС создает целый ряд преимуществ в использовании вычислительной техники, которые невозможно или трудно обеспечить при использовании одной микро-, мини-ЭВМ или даже большой ЭВМ.

Во-первых, ЛВС предоставляет возможность применения отдельных дорогостоящих устройств, подключенных к одной из микроЭВМ сети, многими пользователями. Такими устройствами могут быть графопостроители планшетного типа для изготовления чертежей большого размера, устройства печати высокого качества для воспроизведения цветных изображений, накопители на магнитных дисках большой емкости и др.

Во-вторых, ЛВС обеспечивает совместное использование банков данных (упорядоченных наборов данных) большого размера, размещаемых, как правило, на НМД большой емкости, всеми пользователями (абонентами) сети. При этом каждому абоненту сети обеспечивается оперативный доступ к информации, хранящейся в банке данных.

В-третьих, ЛВС позволяют реализовать безбумажную технологию обработки документации. Такая обработка заключается в том, что документы, совместно используемые абонентами сети, хранятся на магнитных носителях, передаются из одной микроЭВМ сети в другую по каналам связи, работа с ними ведется на экране дисплея и только в некоторых случаях тексты документов выводятся на печать.

В-четвертых, ЛВС обеспечивают оперативный обмен текстами программ и другой управляющей информацией между отдельными микроЭВМ сети, что особенно важно при реализации ГАП.

К настоящему времени сложились основные подходы к разработке и реализации ЛВС микроЭВМ. При создании ЛВС стремятся ограничиться простой структурой связей, компенсируя простоту организации сети высокой скоростью обмена данными между отдельными абонентами сети. Можно выделить три наиболее используемые конфигурации ЛВС (рис. 11.1): магистральную (шинную), кольцевую и звездообразную.

В сетях с магистральной структурой данные, передаваемые любой микроЭВМ, равнодоступны для остальных микроЭВМ. В сетях с кольцевой структурой данные передаются в одном направлении, указанном стрелкой на рис. 11.1 *б*, к соседней микроЭВМ, через которую они транслируются в следующую микроЭВМ, и т. д. В звездообразной конфигурации в качестве центрального узла (узла коммутации) может служить микро- или мини-ЭВМ, осуществляющая коммутацию данных между микроЭВМ сети.

В качестве абонентов ЛВС могут использоваться микроЭВМ различных классов, в том числе и персональные ЭВМ. Набор внешних устройств, входящих в состав микроЭВМ сети, зависит от функций, возлагаемых на данного абонента сети. Например, если в качестве абонента сети используется

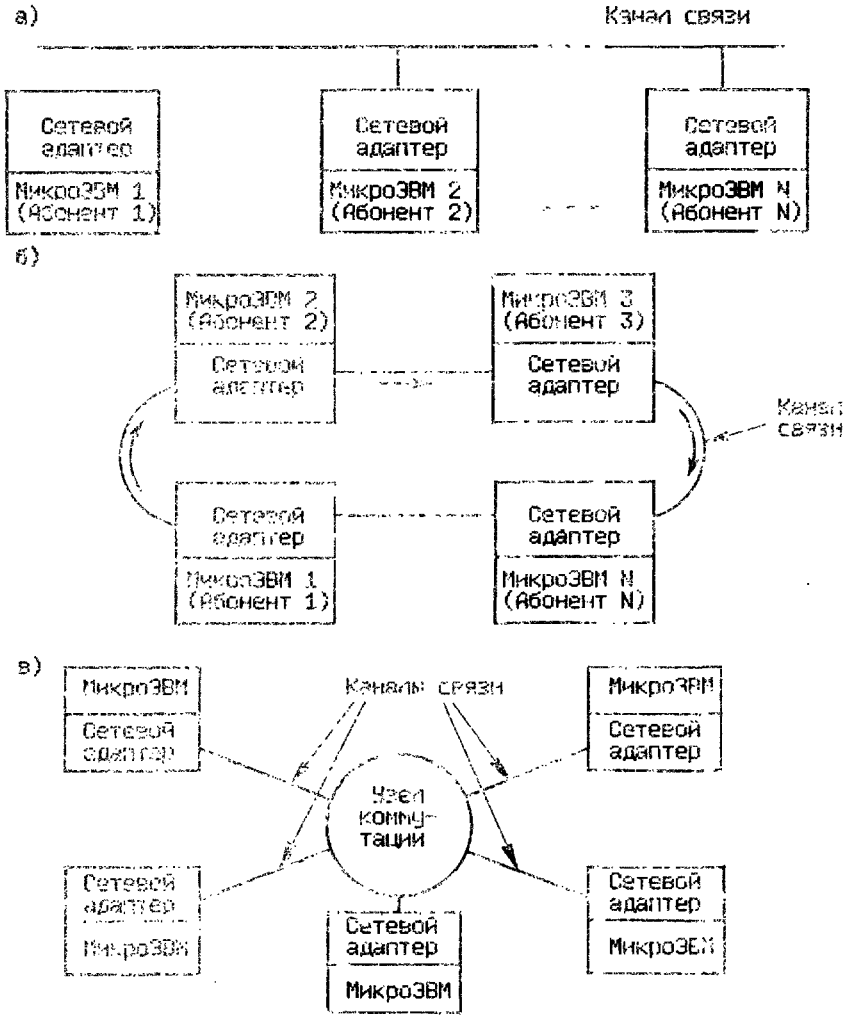


Рис. 11.1. Структуры локальных вычислительных сетей микроЭВМ:
а — магистральная; *б* — кольцевая; *в* — звездообразная

встраиваемая микроЭВМ, то в состав ее внешних устройств будут входить только устройства сопряжения с объектом. Если же в качестве абонента сети служит автономная микроЭВМ или персональная ЭВМ, то ее обычно оснащают дисплеем, устройством печати, накопителями на гибких или жестких магнитных дисках.

МикроЭВМ и персональные ЭВМ при такой комплектации называют рабочими станциями ЛВС. Некоторые рабочие станции ЛВС оснащают дополнительными устройствами, предназначенными для совместного использования всеми абонентами сети. Однако любая микроЭВМ сети должна иметь в своем составе устройство сопряжения с сетью — сетевой адаптер. Следует отметить, что первые модели автономных микроЭВМ и персональных ЭВМ разрабатывали без учета их возможного использования в ЛВС и не оснащали сетевыми адаптерами. В последних моделях таких микроЭВМ сетевой адаптер является обязательным устройством.

В качестве канала связи в ЛВС используется, как правило, последовательный интерфейс — наиболее эффективное средство связи при малых расстояниях между микроЭВМ, характерных для локальных сетей. В последовательных интерфейсах для передачи электрических сигналов применяют скрученные пары проводов, коаксиальные и волоконно-оптические кабели, которые при расстояниях порядка 10^3 м обеспечивают высокую пропускную способность канала связи — $1—10^2$ Мбит/с. При такой высокой пропускной способности для передачи данных между всеми микроЭВМ сети достаточно иметь единственный канал связи — моноканал, обслуживающий все микроЭВМ сети в режиме мультиплексирования. Именно на основе моноканала и строятся ЛВС двух основных конфигураций — магистральные и кольцевые.

Завершая краткое рассмотрение локальных вычислительных сетей микроЭВМ, следует отметить, что только широкое внедрение ЛВС микроЭВМ обеспечит действительное ускорение научно-технического прогресса в основных отраслях народного хозяйства. Более подробно с различными аспектами организации и применения ЛВС микроЭВМ и персональных ЭВМ можно ознакомиться в работе [46].

11.6. Примеры применения микроЭВМ «Электроника-60»

Рассмотрим два примера применения микроЭВМ «Электроника-60» для управления работой внешних устройств, не входящих в состав стандартных ВУ микроЭВМ: в первом показан процесс разработки нестандартного контроллера ВУ, а во втором иллюстрируется использование стандартных функциональных блоков микроЭВМ.

Пример 11.1. Система функционального моделирования цифровых устройств на основе микроЭВМ «Электроника-60» и пульта управления и индикации (ПУИ).

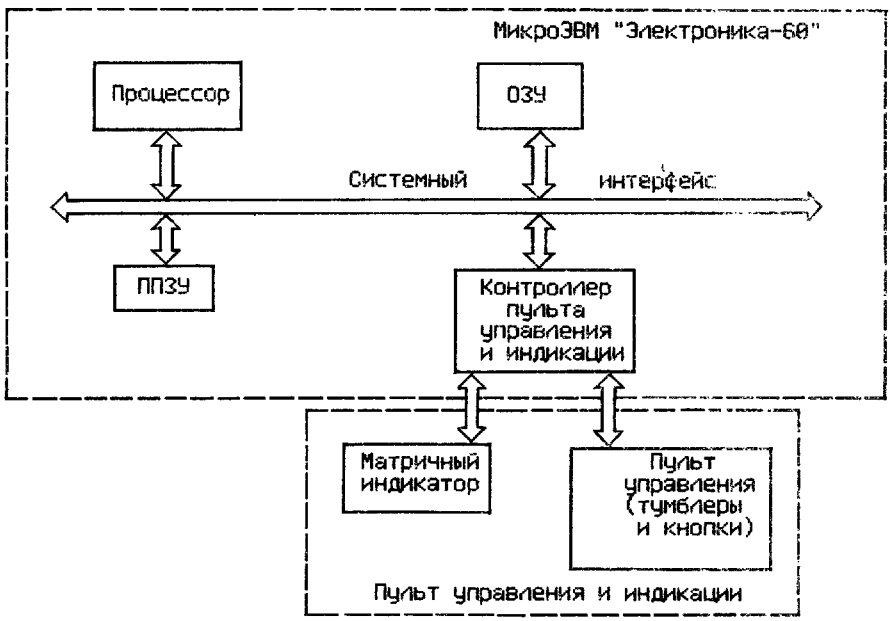


Рис. 11.2. Структурная схема системы функционального моделирования на основе микроЭВМ «Электроника-60»

МикроЭВМ, к которой в качестве нестандартного внешнего устройства подключается ПУИ (см. параграфы 9.2 и 10.6), выполняет следующие функции:

принимает информацию от тумблеров и кнопок пульта управления;

реализует программы моделирования цифровых устройств;

обеспечивает вывод результатов моделирования на матричный индикатор.

Для выполнения возложенных на микроЭВМ функций в ее состав достаточно включить следующие функциональные блоки (рис. 11.2):

процессор;

перепрограммируемое запоминающее устройство (ППЗУ);

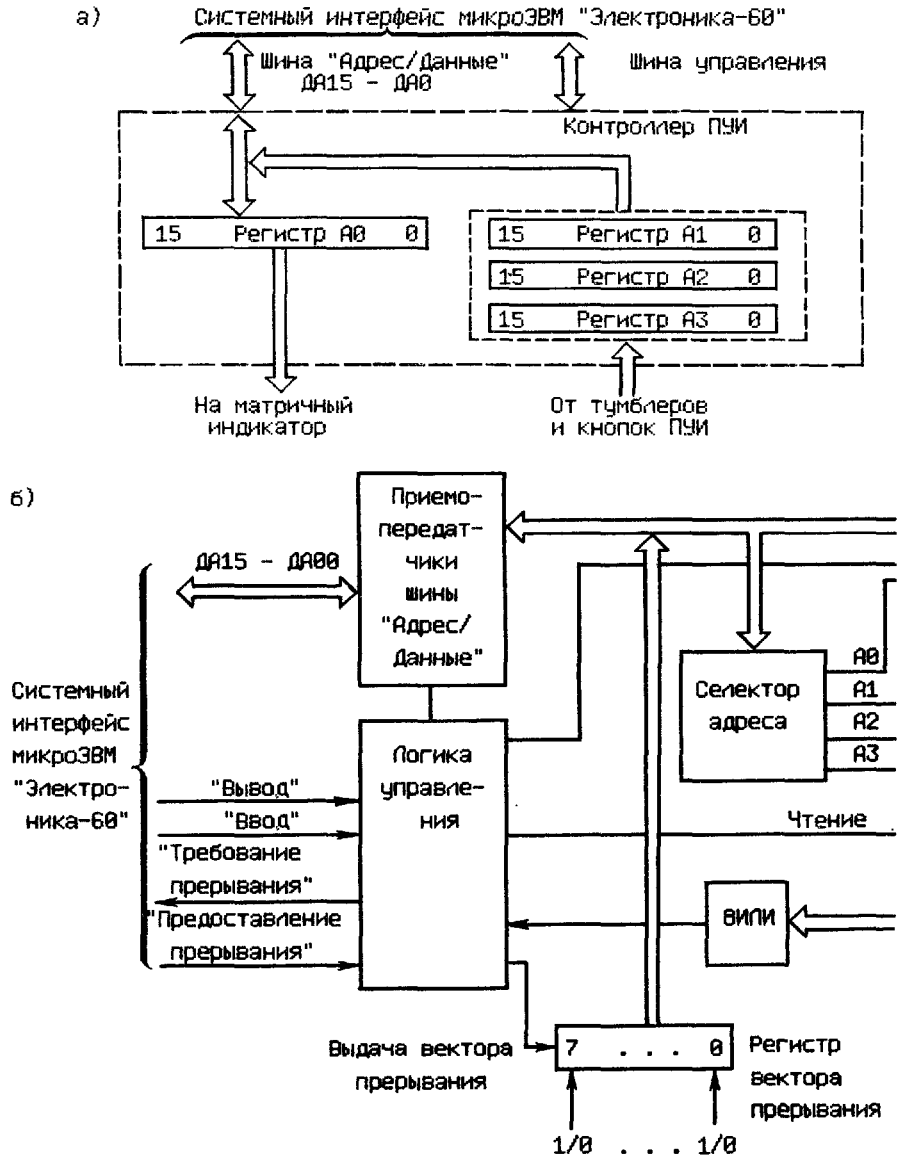


Рис. 11.3. Контроллер ПУИ:

а — программная модель; *б* — структурная схема

оперативное запоминающее устройство (ОЗУ);

контроллер пульта управления и индикации.

Здесь единственным нестандартным модулем является контроллер пульта управления и индикации. Рассмотрим подробнее его функциональную и структурную организацию.

Как было показано в параграфе 10.6, контроллер должен обеспечивать следующие режимы обмена данными микроЭВМ с пультом управления и индикации:

прием с пульта управления информации, задаваемой тумблерами и кнопками, и передачу ее в микроЭВМ как в асинхронном режиме обмена, так и в режиме прерывания программы;

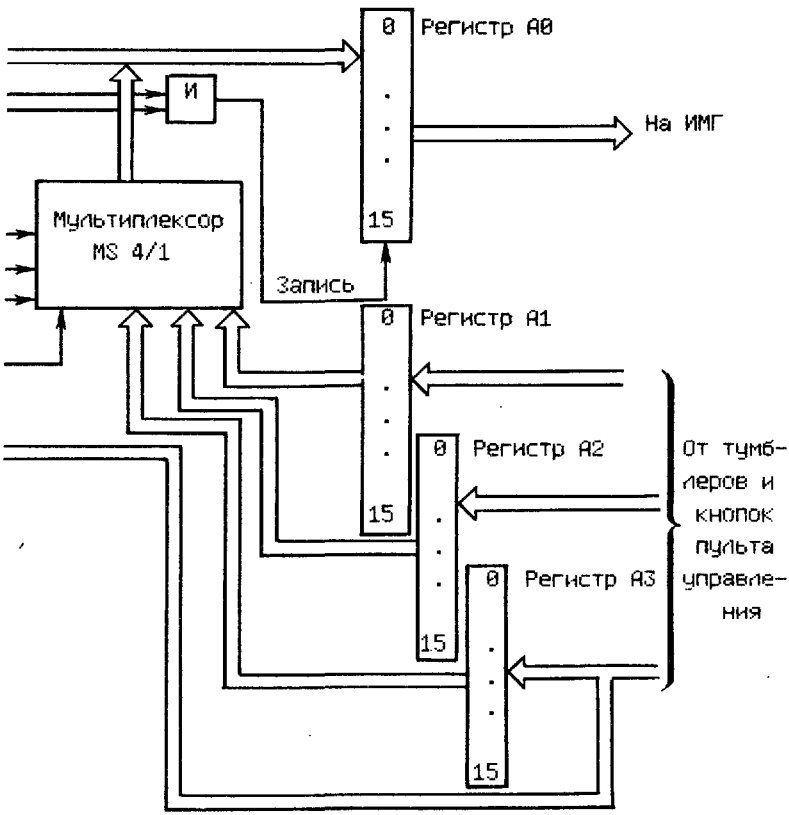
передачу на индикатор управляющей информации из микроЭВМ в синхронном режиме обмена.

Основное влияние на структурную организацию контроллера ПУИ оказывает реализованный в микроЭВМ «Электроника-60» порядок обмена данными с внешними устройствами. МикроЭВМ «Электроника-60» построена на основе системного интерфейса с общей адресно-информационной шиной (см. параграфы 5.2, 8.1 и рис. 5.2, *г*), при использовании которого обмен информацией с ВУ организуется по аналогии с обращениями к памяти.

Как правило, обмен информацией в микроЭВМ «Электроника-60» производится 16-разрядными словами, поэтому в контроллере ПУИ необходимо реализовать, как минимум, четыре 16-разрядных регистра. Через три регистра информация от 40 тумблеров и восьми кнопок пульта управления * будет передаваться в микроЭВМ, а четвертый необходим для передачи управляющей информации из микроЭВМ на матричный индикатор. Организация обмена информацией с регистрами контроллера ПУИ по аналогии с обращениями к памяти требует, чтобы каждому из четырех регистров контроллера был присвоен индивидуальный адрес из

---

* Здесь мы намеренно не обсуждаем функциональное назначение отдельных тумблеров и кнопок пульта. Это объясняется тем, что функциональное назначение сигналов от тумблеров и кнопок определяется типом моделируемого цифрового устройства и влияет только на алгоритмы обработки (моделирования), но не на структурную организацию контроллера ПУИ.



области адресов микроЭВМ, отведенных для ВУ. Все это позволяет построить программную модель контроллера ПУИ (рис. 11.3, *а*), на которой отображены следующие его регистры:

A0 — регистр для передачи информации на матричный индикатор;

A1, A2, A3 — регистры для приема информации от тумблеров и кнопок пульта управления.

Распределение по регистрам *A0*, ..., *A3* информации, поступающей от пульта управления в микроЭВМ и пересылаемой из микроЭВМ на индикатор, обеспечивает возможность реализации всех требуемых режимов обмена информацией микроЭВМ с ПУИ. Это можно сделать, например, с помощью программ, аналогичных программам из примеров 8.1 и 8.2.

Структурная схема контроллера ПУИ приведена на рис. 11.3, б. Принципы работы контроллеров такого типа достаточно подробно рассмотрены в параграфах 8.3 и 8.5. Поэтому здесь обратим внимание только на некоторые особенности структурной организации контроллера ПУИ. Наличие в контроллере четырех регистров данных потребовало включения в его схему селектора адреса, формирующего четыре адресных сигнала *A0* ..., *A3*. Адресный сигнал *A0* совместно с управляющим сигналом системного интерфейса «Вывод» обеспечивает запись информации в выходной регистр *A0*, из которого она сразу поступает на матричный индикатор. Адресные сигналы *A1*, *A2* и *A3* управляют работой мультиплексора, который обеспечивает подключение к шине данных системного интерфейса выходов одного из регистров данных *A1*, *A2* или *A3* соответственно.

Управляющий сигнал «Требование прерывания» системного интерфейса формируется при нажатии на одну из кнопок пульта управления следующим образом. Сигналы от кнопок пульта управления параллельно с подачей на восемь старших разрядов регистра *A3* поступают на восьмивходовую схему ИЛИ. Следовательно, при нажатии любой кнопки на ПУИ на выходе схемы ИЛИ контроллера появляется сигнал, из которого логика управления контроллера сформирует управляющий сигнал системного интерфейса «Требование прерывания».

Контроллер ПУИ, реализованный на ИМС 155 и 589 серий, свободно размещается на одной стандартной плате микроЭВМ «Электроника-60» размерами 252×148 мм.

Перепрограммируемое запоминающее устройство микроЭВМ используется для хранения всего комплекса программ системы функционального моделирования, включающего и программы обмена информацией с пультом управления и индикации. Применение в системе функционального моделирования перепрограммируемого ЗУ, во-первых, обеспечивает оперативную переналадку системы (замена платы или БИС ППЗУ) на моделирование различных типов цифровых устройств и, во-вторых, исключает необходимость комплектования микроЭВМ системы моделирования внешним устройством для ввода программ.

Оперативное запоминающее устройство микроЭВМ используется в качестве буферного ЗУ при обмене информацией с пультом управления и индикации и для хранения векторов прерываний и массивов данных, модифицируемых при моделировании.

Процессор, ППЗУ и ОЗУ являются стандартными модулями микроЭВМ «Электроника-60» и выбираются из разработанного для нее набора стандартных модулей. Конкретные типы модулей ППЗУ и ОЗУ выбираются исходя из объемов памяти, требуемой для размещения программ моделирования и другой служебной информации (векторов прерываний, массивов данных и т. п.).

Ниже приведен фрагмент программы обработки прерывания от кнопок пульта управления и ввода информации из тумблерного регистра «Адрес/Данные» пульта управления при моделировании процессов функционирования базовой ЭВМ.

```
WWDA:MOV R1,-(R6) ;   сохранение в стеке ОЗУ
     MOV R2,-(R6) ;содержимого регистров R1 и R2
     MOV @#A3,R1  ;чтение в регистр R1 процессора содержимого
                  ;регистра A3 контроллера
     BMI M8       ;переход, если старший разряд регистра A3 контроллера
                  ;троллера равен 1, т.е. нажата кнопка Nº 8 на ПУИ
       ...

  M8:MOV @#A2,R2  ;    чтение в регистр R2 процессора содержимого
                  ;регистра A2 контроллера (регистр "Адрес/Данные" ПУИ)
     MOV R2,@#ADR ;пересылка содержимого регистра R2 в ячейку памяти,
                  ;    предназначенную для размещения информации
                  ;            от регистра "Адрес/Данные"
```

```
MOV (R6)+,R2 ; восстановление содержимого
MOV (R6)+,R1 ;регистров R1 и R2 процессора
RTI          ;возврат из подпрограммы обработки прерываний
```

Пример 11.2. Установка для обработки изображений на основе микроЭВМ «Электроника-60».

Рассмотрим сначала функциональное назначение и принципы работы установки в целом. Установка предназначена для преобразования в цифровую форму представления и обработки изображений размерами не более чем 300×300 мм, полученных на прозрачных носителях (фотопластинках, фотопленках и т. п.). Обработка изображений, представленных в цифровой форме, может включать, например, вычисление плотности изображения в отдельных точках или областях изображения, построение линий одинаковой плотности изображения (изофот) или линий с плотностью изображения, изменяющейся по заданному закону.

Принципиальная схема измерений плотности изображения в отдельных точках для последующего преобразования в цифровую форму представлена на рис. 11.4.

Установка состоит из измерительной части, на которую возлагается задача преобразования плотности изображения в отдельной точке в аналоговый сигнал, и микроЭВМ. МикроЭВМ обеспечивает:

преобразование аналогового сигнала от фотоприемника в цифровую форму и запись полученного значения в память;

управление перемещением стола с закрепленным на нем носителем изображения;

обработку изображения (либо его части), записанного в цифровой форме в память.

Стандартные комплектации микроЭВМ «Электроника 60» не обеспечивают выполнения перечисленных выше функций. Для определения необходимого состава функциональных блоков микроЭВМ рассмотрим сначала параметры сигналов (формат и язык), которыми микроЭВМ должна обмениваться с измерительной частью установки.

Фотоприемник формирует сигнал постоянного напряжения в диапазоне 0...5 В в зависимости от плотности изображения в отдельной точке. Следовательно, для приема информации о плотности изображения и последующего ее преобразования в цифровую форму микроЭВМ должна быть оснащена аналого-цифровым преобразователем (см. параграф 7.3). Перечень стандартных модулей микроЭВМ «Электроника 60» включает модуль АЦП, который обеспечивает преобразование исходного аналогового сигнала в 12-разрядный код за время, не превышающее 50 мкс.

Подвижный стол, на котором закрепляется носитель изображения, может перемещаться в горизонтальной плоскости относительно неподвижных осветителя и фотоприемника. Тем самым обеспечиваются измерения плотности изображения в отдельных точках носителя. Система управления движением стола обеспечивает прямолинейное перемещение стола в прямом и обратном направлениях по осям X и Y или по любой траектории, заданной функцией двух перем-

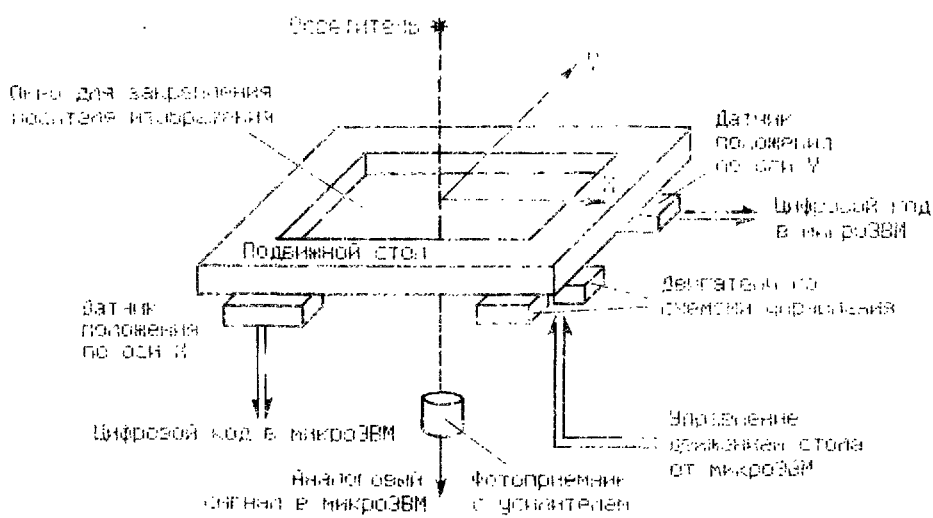


Рис. 11.4. Схема измерительной части установки для обработки изображений

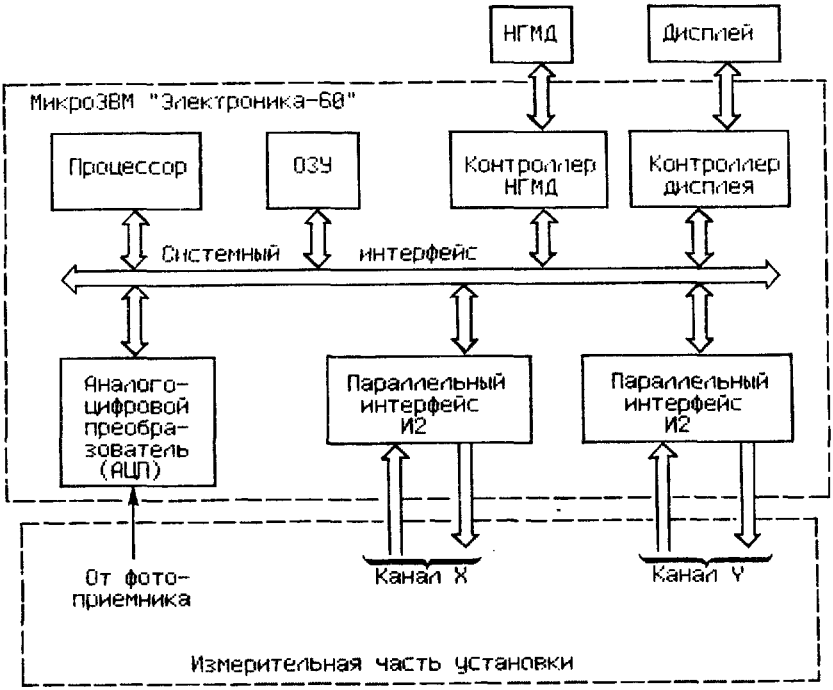


Рис. 11.5. Структурная схема микроЭВМ установки для обработки изображений

менных X и Y. Заданные перемещения стола осуществляются с помощью двух двигателей со схемами управления и двух датчиков положения, измеряющих величину перемещения по осям X и Y с точностью до 5 мкм. Датчики вырабатывают 16-разрядный двоичный код величины и направления перемещения. При этом старший разряд кода указывает направление перемещения (знак «+» или «–»), а остальные 15 разрядов — его величину (0—163,84 мм). Для ввода в микроЭВМ информации от двух датчиков положения (X и Y) необходимо оснастить ее двумя 16-разрядными параллельными интерфейсами ввода.

Управление работой двигателей производится с помощью трех управляющих сигналов: «Включить двигатель», «Выключить двигатель» и «Направление движения». Следовательно, микроЭВМ должна формировать и передавать в схему управления каждым двигателем трехразрядный двоичный код, соответствующий заданному перемещению стола. Решить эту задачу можно с помощью одного шестиразрядного параллельного интерфейса либо двух трехразрядных. Время работы двигателей, а следовательно, и величина перемещения стола отслеживаются программным путем по информации, поступающей от датчиков положения. Поскольку информация о перемещении стола поступает в микроЭВМ по двум раздельным каналам (X и Y), то и сигналы управления на двигатели целесообразно передавать по раздельным интерфейсам вывода. Таким образом в установке реализуются два независимых канала управления перемещением стола, каждый из которых включает параллельный интерфейс ввода-вывода, двигатель и датчик положения.

В качестве параллельного интерфейса, обеспечивающего работу одного канала управления, может быть использован стандартный модуль микроЭВМ «Электроника-60» параллельный интерфейс *И2*, который обеспечивает ввод и вывод 16-разрядных слов информации по раздельным шинам. Всего для комплектации установки потребуются два параллельных интерфейса *И2*. Структурная схема микроЭВМ «Электроника-60», входящей в установку для обработки изображений, приведена на рис. 11.5.

Конкретный набор стандартных внешних устройств (накопители на магнитных дисках и лентах, устройства печати, графопостроители и т. д.) для микроЭВМ, входящую в установку для обработки изображений, определяется исходя из требований к объемам хранимой информации (цифрового представления изображения), сложности программ обработки, форм документирования результатов работы и т. п.

Глава 12
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МИКРОЭВМ

12.1. Общие сведения

Программное обеспечение современных микроЭВМ чрезвычайно разнообразно, и его трудно подробно рассмотреть в рамках данной книги. Поэтому здесь дан краткий обзор этого важнейшего компонента любой управляющей и информационно-вычислительной системы с упором на разъяснение терминологии, используемой в литературе по микроЭВМ.

Ценность микроЭВМ, не имеющей программ, соизмерима с ценностью телевизора, привезенного в местность, удаленную от телецентров и ретрансляторов телевизионных программ. В связи с этим при выборе того или иного типа микроЭВМ для встраивания в аппаратуру, использования в системе сбора и обработки информации, управления ходом эксперимента, создания обучающей системы для школьников или студентов и других применений следует в первую очередь выяснить, существуют ли для данного типа микроЭВМ:

комплекс готовых программ, часть из которых можно было бы прямо или после доработки использовать для решения поставленной задачи;

развитая система программных и аппаратных средств для модернизации существующих или создания новых программ и их отладки.

При этом надо учитывать, что наиболее трудоемким этапом процесса разработки программы является не кодирование, которое некоторые пользователи и авторы книг ошибочно называют программированием, а проектирование программы (алгоритмизация) и ее отладка (рис. 12.1). Уменьшения же трудоемкости этапа кодирования добились за счет создания различных языков программирования: языка ассемблера, алгоритмических языков высокого уровня (БЕЙСИК, ФОРТРАН, ПАСКАЛЬ и т. п.), а также проблемно-ориентированных языков, приближающихся к естественному профессиональному языку тех или иных специалистов (медиков, химиков, шахматистов и т. п.).

Язык ассемблера — это наиболее простой язык, который уже использовался в тексте книги (язык мнемонического кодирования). Мнемонические имена команд облегчают составление и чтение программ (не надо запоминать и расшифровывать последовательности нулей и единиц или шестнадцатеричных цифр). Использование же символических адресов (имен переменных) освобождает программиста от утомительной работы по распределению памяти ЭВМ для переменных и констант. Правда, последнего можно добиться только при машинном переводе мнемонической программы на язык ЭВМ, когда такой перевод (ассемблирование) осуществляется с помощью программы, называемой ассемблером. Ассемблер преобразует мнемонические коды в двоичные коды команд, строит таблицы имен и их значений и заменяет все ссылки на имена соответствующими двоичными числами. Очевидно, что ЭВМ может ассемблировать программы быстрее и точнее программиста.

Ассемблер — это программа, воспринимающая в качестве входных данных исходную программу или исходный модуль (текст программы или подпрограммы на языке ассемблера) и выдающая на выходе объектную программу или объектный модуль (программу или подпрограмму на машинном языке, дополненную информацией для осуществления связи этого модуля с

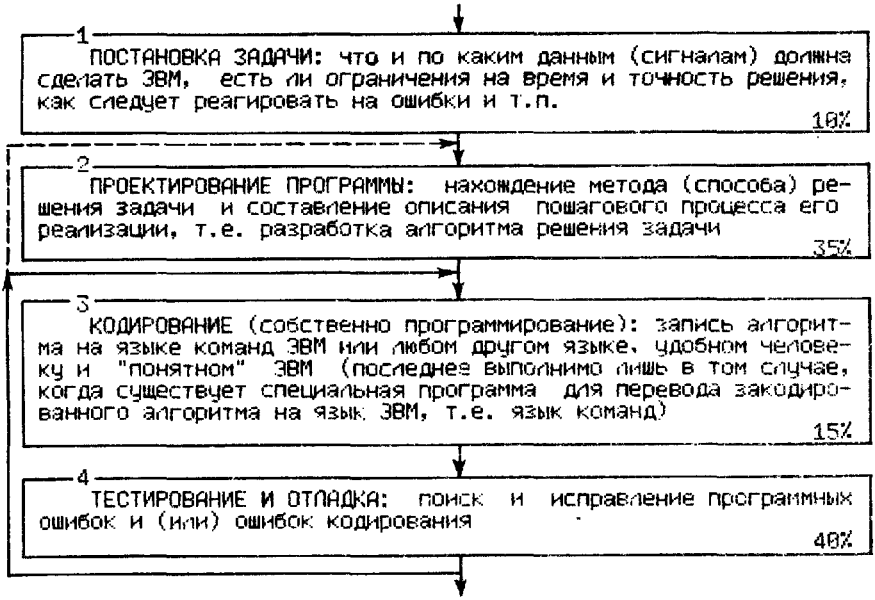


Рис. 12.1. Этапы разработки программ и их относительная трудоемкость (в %)

другими и загрузки в требуемое место памяти ЭВМ). Аналогичные функции выполняют трансляторы с языков высокого уровня (об этом было сказано в параграфе 4.1). Однако такие процедурно-ориентированные языки не учитывают особенностей конкретной ЭВМ и их можно использовать для составления программ, которые переводятся на язык любой ЭВМ (если эта ЭВМ снабжена соответствующим транслятором). Программы же, написанные на языке ассемблера для одного типа ЭВМ, нельзя преобразовать в последовательность команд для ЭВМ другого типа, так как в таких программах учитываются конкретная архитектура, система команд и методы адресации. Но зато только на языке ассемблера (если не считать языка команд ЭВМ) можно составить программу, имеющую минимальный объем и максимальное быстродействие, что очень важно для встраиваемых микроЭВМ.

Для одного и того же типа ЭВМ может существовать несколько ассемблеров, различающихся:

набором псевдокоманд и директив, предназначенных для задания имени и указания начала и конца программного модуля, управления трансляцией, описания переменных и констант, резервирования памяти, указания эквивалентных величин и т. п.;

возможностью использования макрокоманд (заменяя при кодировании одинаковые последовательности команд макрокомандой, пользователь освобождается от утомительной работы по многократному переписыванию команд последовательности и «поручает» такую работу ассемблеру, который при наличии средств для работы с макрокомандами носит название макроассемблера);

числом определяемых имен, скоростью ассемблирования, способами идентификации ошибок, возможностью создания переместимых программ (программ, которые могут загружаться в любое место памяти ЭВМ), количеством проходов при трансляции (в более простых двухпроходных ассемблерах окончательный текст объектной программы создается лишь после повторного чтения исходной программы) и т. д.

Естественно, что ассемблеры, предоставляющие наилучшие средства для ускорения написания программ (макроассемблеры), занимают наибольшие объемы памяти и чаще всего не могут быть размещены в памяти тех микроЭВМ, для которых предназначены объектные программы, получаемые на их выходе. Поэтому программы, написанные, например, на языке ассемблера для микропроцессора К580, могут транслироваться либо на инструментальных ЭВМ (системах для разработки и проверки аппаратных и программных средств однотипных микроЭВМ), либо на любой другой ЭВМ с развитым набором внешних устройств, если для нее написана программа кросс-ассемблера с языка ассемблера микропроцессора К580.

В процессе разработки ассемблерных программ для микроЭВМ пользователю кроме ассемблера или кросс-ассемблера потребуется несколько других программ (рис. 12.2). Это прежде всего редактор текста, позволяющий вводить в машину и редактировать отдельные модули программы на языке ассемблера (головную программу и подпрограммы). Результатом работы редактора текста является исходный модуль — последовательность кодов символов, которая может быть оставлена в памяти для последующего ассемблирования или сохранена, например, на дискете НГМД.

Загрузив теперь в машину программу АССЕМБЛЕР и исходный модуль (если он не сохранился в памяти после этапа ввода и редактирования), можно получить объектный модуль. Кроме объектного модуля ассемблер позволяет получить все или некоторые из перечисленных ниже документов:

листинг (распечатку информации о результатах ассемблирования, содержащую программу в исходном и объектном коде);

список ошибок ассемблирования;

таблицу символических имен, используемых в программе;

таблицу перекрестных ссылок, содержащую перечень имен, и список всех команд, в которых они используются;

список внешних ссылок (перечень имен подпрограмм или переменных, которые определены за пределами данной программы, т. е. являются глобальными переменными);

список подпрограмм или макрокоманд с указанием их длины.

По требованию ассемблер записывает объектный модуль на дискету или выводит на перфоленту.

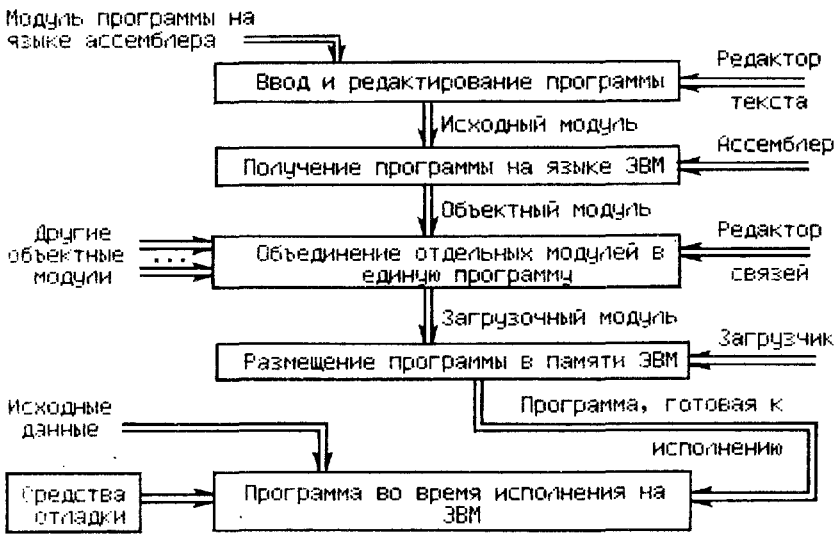


Рис. 12.2. Этапы создания и выполнения ассемблерных программ

В развитых системах программного обеспечения микроЭВМ существуют системные библиотеки, содержащие объектные модули распространенных подпрограмм. К ним относятся подпрограммы перевода чисел из одной системы счисления в другую, перекодировки, вычисления некоторых функций, выполнения операций ввода-вывода с использованием стандартных устройств ввода-вывода (таймера, АЦП, ЦАП и т. д.), называемые драйверами ввода-вывода, и ряд других подпрограмм. Поэтому практически любую (даже маленькую) программу целесообразно собирать из нескольких подпрограмм, которые должны быть взяты из библиотеки и (или) написаны пользователем. Объединение объектных модулей этих подпрограмм осуществляется редактором связей (компоновщиком).

Выход редактора связей называется загрузочным модулем. Если загрузочный модуль записывается для хранения, например, на дискету, то для загрузки его в память и подготовки к исполнению используется еще одна программа, называемая загрузчиком. Иногда связывание и загрузка объединяются в одну программу (связывающий загрузчик). Конечным результатом всего этого процесса является программа, которая загружена в память и готова к выполнению.

Тестирование и отладка предназначены обеспечить правильность функционирования разработанной программы. Цель тестирования заключается в выявлении в программе ошибок. Цель отладки состоит в установлении причин, вызывающих ошибки, выявленные при тестировании, и в их устранении. Это очень трудоемкий этап процесса разработки программ (см. рис. 12.1). Подробнее он рассмотрен в параграфе 12.2 - 12.4.

Аналогичны этапы разработки программ, кодируемых на языках высокого уровня. Изменяется лишь программа, с помощью которой производится преобразование исходных модулей в объектные модули (например, вместо ассемблера используется компилятор с языка ФОРТРАН). Так как структура объектного модуля, как правило, не зависит от языка, на котором написан исходный модуль, то отдельные исходные модули программы могут быть написаны на разных языках программирования.

Кроме перечисленных программных средств, которые обычно входят в состав операционной системы, при подготовке программ для микроЭВМ используются моделирующие программы, эмуляторы и ряд других программ.

12.2. Программное обеспечение микроконтроллеров и встраиваемых микроЭВМ

Безусловное выполнение всех запланированных функций при небольших габаритах, малых затратах энергии и низкой стоимости - вот основные требования, предъявляемые к микроконтроллерам и микроЭВМ, встраиваемым в те или иные изделия. Причем чем больше выпуск таких изделий, тем важнее становятся последние из перечисленных требований:

уменьшение энергозатрат каждого из изделий, выпускаемых массовым тиражом, позволяет существенно сэкономить энергетические ресурсы страны;

снижение стоимости микроконтроллера или встраиваемой микроЭВМ уменьшает себестоимость изделия и позволяет снизить его цену и (или) получить бо́льшую прибыль.

Обеспечить же снижение энергозатрат и стоимости микроконтроллера или микроЭВМ можно за счет не только подбора оптимальных аппаратных средств, но и оптимизации программы их работы. Так как в микроконтроллерах и во встраиваемых микроЭВМ программа записывается в ПЗУ (см. параграф 6.3), то сокращение программы позволяет уменьшить объем ПЗУ и либо сократить количество БИС ПЗУ, либо использовать в нем БИС с мень-

шей емкостью (меньшими энергозатратами и стоимостью). Это предопределяет наивысшие требования к процессу разработки программ для микроконтроллеров и микроЭВМ, встраиваемых в массовые изделия.

Но как же отладить и сравнить различные варианты программ, если единственный путь ввода программы в микроконтроллер или во встраиваемую микроЭВМ — это наличие в них ПЗУ с программой, записанной на специальной установке? Не идти же по пути выбрасывания ПЗУ с неудачным вариантом программы, записи в «чистое» ПЗУ нового варианта программы и т. д.? Да и как проверять эти варианты на ЭВМ, внешние устройства которой могут лишь получать сигналы из изделия и управлять (правильно или неправильно) его работой?

Поскольку сам микроконтроллер или встраиваемую микроЭВМ (будем называть их прототипом) трудно приспособить для создания собственного программного обеспечения, то используют специальные средства разработки на универсальных ЭВМ или инструментальных ЭВМ.

Инструментальная ЭВМ — это достаточно мощная система, построенная на микроЭВМ, имеющей или эмулирующей архитектуру прототипа (термин «эмуляция» разъяснялся в конце параграфа 7.4). Хорошая инструментальная ЭВМ содержит:

дисковое ЗУ, сохраняющее большое число системных и прикладных программ, используемых при проектировании и отладке прототипа;

большую оперативную память для размещения монитора (управляющего работой инструментальной ЭВМ), системных и (или) прикладных программ (непосредственно исполняемых на тех или иных этапах работы), моделирования памяти прототипа и т. д.);

средства, позволяющие проектировщику вызвать нужные программы, установить режим их выполнения (например, по командам или тактам), дать необходимую команду на прототип или его модель и ряд других приказов;

средства для индикации работы инструментальной ЭВМ и связанного с ней прототипа (обычно дисплеи);

контроллеры для подключения печатающих устройств, графопостроителей, устройств ввода-вывода с перфоленты, программаторов ППЗУ и других устройств для документирования и вывода информации;

аппаратуру для подключения прототипа и (или) его отдельных частей, а также управления ими;

систему ввода-вывода аналоговых сигналов для общения со средой (моделью среды), в которой будет работать прототип.

Кроме этих (а нередко и ряда других) аппаратных средств инструментальная ЭВМ снабжена пакетами служебных и прикладных программ. Среди этих программ обязательно существуют хотя бы один редактор текста для создания и модификации текста программ (вариантов программ для прототипа или любых отладочных программ, разрабатываемых в процессе проектирования), несколько трансляторов, редактор связей, загрузчик и набор других служебных программ (например, программы для отладки в реальном масштабе времени).

Выше оговаривались причины, обусловливающие необходимость создания минимальных по объему программ для встраиваемых ЭВМ. Поэтому среди трансляторов инструментальных ЭВМ обязательно есть ассемблеры. Однако при начальной отработке программ прототипа и для создания отладочных программ могут использоваться языки высокого уровня и инструментальные ЭВМ снабжаются также интерпретаторами и компиляторами (см. параграф 4.1) языков БЕЙСИК, ФОРТРАН, ПАСКАЛЬ, С и рядом других (например, языка ПЛ/М, разработанного для написания служебных и прикладных программ микропроцессоров К580).

Кроме помощи в создании программного обеспечения прототипа хорошая инструментальная ЭВМ позволяет произвести совместную наладку аппаратных и программных средств. Для этого используется эмулятор — мощное отладочное средство, заменяющее дорогие диагностические устройства. Эмулятор заменяет микропроцессор прототипа и выполняет команды микропроцессора в среде прототипа под управлением инструментальной ЭВМ. Поэтому при отладке прототип может использовать бо́льшую часть аппаратных и программных ресурсов инструментальной ЭВМ. Программу можно выполнять по командам и индицировать состояния устройств, аппаратные средства прототипа (модули памяти, контроллеры ввода-вывода, схемы управления прерываниями) можно проверить и отладить индивидуально с помощью программ инструментальной ЭВМ.

Перечисление достоинств инструментальных ЭВМ можно было бы продолжить, но следует упомянуть и о их недостатках. Большинство таких ЭВМ позволяет производить наладку прототипов, построенных на БИС МПК одной серии. Поэтому пользователи, работающие с разными МПК и создающие на их базе встраиваемые микроЭВМ, должны иметь несколько инструментальных ЭВМ разных типов или очень мощную (а следовательно, дорогую) систему, заменяющую эти машины. Такой путь оправдан лишь для организаций, специализирующихся на разработке различных встраиваемых микроЭВМ и (или) занимающихся их наладкой. Однако даже эти организации часто подготавливают программы прототипа на универсальных ЭВМ.

Универсальные ЭВМ любого типа (большие ЭВМ, миниЭВМ или персональные ЭВМ) часто снабжают кросс-ассемблерами и имитаторами для подготовки и отладки программ многих микропроцессорных систем.

Кросс-ассемблер выполняет точно такие же функции, что и самоассемблер: трансляция мнемонических команд в машинный код, выполнение псевдокоманд и директив, обнаружение синтаксических ошибок и обработка макрокоманд. Кросс-ассемблер реализуется на ЭВМ, имеющей другую систему команд, отличную от микропроцессора, для которого создается ассемблированная программа (т. е. универсальная ЭВМ преобразует мнемонические команды не в свои машинные команды, а в чужие — команды микропроцессора). Естественно, что при подготовке текста программы на языке ассемблера и его редактировании могут использоваться все необходимые средства универсальной ЭВМ: редакторы текста, печатающие устройства, дисковые ЗУ (для сохранения созданного исходного текста и полученной объектной программы) и т. п.

Моделирующие программы предназначены для отладки объектных программ, полученных после их ассемблирования с помощью кросс-ассемблера. В этих программах все регистры разрабатываемой микроЭВМ и ячейки ее памяти представлены ячейками памяти универсальной ЭВМ. Моделирующая программа имитирует выполнение каждой команды объектной программы с соответствующим преобразованием этих ячеек. Процессом выполнения программы можно управлять с помощью клавиатуры дисплея, на экран которого выводится вся необходимая для отладки информация. Подробнее о процессе построения моделирующих программ и работе с ними было рассказано в гл. 9.

12.3. Программное обеспечение автономных микроЭВМ

Микроконтроллеры и встраиваемые микроЭВМ целесообразно использовать для управления (поднятия интеллекта) устройств и систем, выпускаемых массовым тиражом. Если же надо автоматизировать работу устройства, выпускаемого небольшими партиями, или обеспечить обработку (в реаль-

ном масштабе времени) данных какого-либо уникального эксперимента, то нецелесообразно затрачивать большие средства и время на разработку встраиваемой микроЭВМ. Дешевле выбрать серийную автономную микроЭВМ (например, микроЭВМ «Электроника-60») и приспособить ее для решения поставленной задачи, даже если ресурсы этой машины будут использованы лишь на 20—30 %. И в первую очередь это относится к пользователям, не имеющим профессиональной подготовки в области вычислительной техники.

Серийные автономные микроЭВМ обычно снабжают неплохими средствами для разработки прикладных программ. Практически все из них имеют хотя бы простейшие средства для связи с пользователем: минимальный набор внешних устройств и ПЗУ с программой, обеспечивающей возможность управления работой микроЭВМ, ввода в нее программ и т. п. Если же для автоматизации приобретена микроЭВМ с НГМД и достаточным объемом оперативной памяти (не менее 16 Кбайт), то работой машины может управлять операционная система (совокупность служебных программ, облегчающих человеку связь с машиной и повышающих эффективность использования ее аппаратных и программных средств).

Операционные системы автономных микроЭВМ — это чаще всего дисковые операционные системы (ДОС), т. е. системы, у которых бо́льшая часть их программ хранится на магнитном диске и вызывается в память микроЭВМ по мере необходимости. В состав ДОС входят:

1) монитор (от лат. monitor — наставник, советник) — программа, управляющая работой ДОС и контролирующая ее функционирование.

Эта программа (или ее основная часть, называемая резидентной) постоянно находится в оперативной памяти машины. Она обеспечивает диалоговый режим связи с микроЭВМ, воспринимая и выполняя команды (приказы), подаваемые с клавиатуры, и выдавая сообщения на дисплей;

2) системные обрабатывающие программы, включающие ассемблер (макроассемблер), компиляторы и интерпретаторы с нескольких языков высокого уровня, библиотеку стандартных программ, редактор текста, редактор связей, загрузчик и т. п.;

3) системные обслуживающие программы, включающие программы диагностики микроЭВМ, отладки создаваемого программного обеспечения; подпрограммы управления вводом-выводом (часто называемые драйверами ввода-вывода); программы обслуживания библиотек и т. п.;

4) генератор ДОС — программа, обеспечивающая создание той урезанной версии ДОС, которая учитывает нужды пользователей микроЭВМ и конкретный состав ее устройств.

Такая версия создается из библиотеки программ, присылаемых разработчиком полной версии ДОС. При генерации в урезанную версию не включается ряд обрабатывающих и (или) обслуживающих программ, которые не нужны пользователям микроЭВМ, что позволяет освободить часть диска для программ пользователей.

Как было отмечено выше, большинство программ ДОС хранится на магнитном диске, на котором кроме них хранятся и программы, создаваемые пользователями микроЭВМ. Общее число программ, записанных на одном диске, может достигать нескольких десятков и даже сотен. Кроме них на диске могут сохраняться и различные данные. Как же определить то свободное место, на которое может быть записана новая программа или новые данные? Как найти на диске ту или иную программу? Как удалить уже не нужную информацию? Всем этим управляет операционная система, используя специальные программы работы с файлами (от англ. file — подшитые бумаги, картотека).

Слово «файл» используется не только в вычислительной технике — английские клерки подшивают письма, приказы и другие документы (данные) в папки с названием «FILE...», аналогичные по виду и назначению нашим папкам с названием «ДЕЛО...». Такая работа проводится для классификации данных по содержанию (жалобы, приказы и т. п.) или по названиям организаций и лиц, от которых они получены. При этом надо всегда помнить, что классификация документов производится для того, чтобы позже их можно было найти по названиям (именам), написанным на папках (файлах).

В вычислительной технике файлом называют совокупность связанных записей, рассматриваемых как единое целое. В файле, содержащем программу, записями являются отдельные операторы или команды этой программы. В файле, содержащем исходные данные к программе, записи — это значения переменных, используемых одной командой ввода. Все файлы, размещенные на одном диске, должны иметь уникальные имена, чтобы их можно было легко найти при работе с микроЭВМ. Операционная система отводит на диске место для каталога файлов и при записи нового файла помещает в этот каталог описание такого файла (имя, дату создания, длину и т. п.). Новые файлы создаются с помощью системных программ редакторов текста (например, текст исходной программы), трансляторов (объектный модуль) и программ пользователей: результаты расчета, эксперимента и т. п.

Программа работы с файлами, входящая в состав ДОС, может (по приказам пользователя) переписывать файлы с одного диска на другой диск (или магнитную ленту), выводить текст каталога или файлов на дисплей или печатающее устройство, объединять несколько файлов, изменять имя файла, устанавливать определенный код защиты файла (запрещая другим пользователям чтение этого файла или работу с ним), удалять файл или группу файлов и т. п.

Существуют мультипрограммные операционные системы, предназначенные для одновременного обслуживания нескольких пользователей. В таких системах есть программы для планирования (диспетчирования) ресурсов микроЭВМ и защиты ее памяти (чтобы программы не могли повлиять друг на друга). Создаются также операционные системы реального времени, представляющие собой мощные мультипрограммные системы, предназначенные для управления сложными производственными процессами. Параллельно со сбором данных, их обработкой и выдачей управляющих воздействий они могут обслуживать нескольких пользователей в режиме разделения времени. Для таких систем характерны жесткие временные ограничения на обработку прерываний от внешних источников. Здесь необходимы планировщик реального времени, библиотека переместимых программ, загрузчик реального времени и достаточно сложный монитор.

Непрерывный рост числа элементов, размещаемых в одной БИС, позволяет записать программы операционной системы в ПЗУ микроЭВМ, что резко улучшает характеристики системы.

12.4. Программное обеспечение персональных ЭВМ

Возможность приспособить цветной телевизор и кассетный магнитофон в качестве дисплея и внешнего ЗУ ЭВМ, а также «упаковать» контроллеры этих устройств, процессор и память ЭВМ в несколько БИС, расположенных внутри клавиатуры, сходной с клавиатурой пишущей машинки, позволили создать к началу 80-х годов очень дешевые микроЭВМ. Такие микроЭВМ были дороже распространенных калькуляторов, но настолько удобнее последних, что их стали покупать вместо программируемых и даже непрограммируемых калькуляторов.

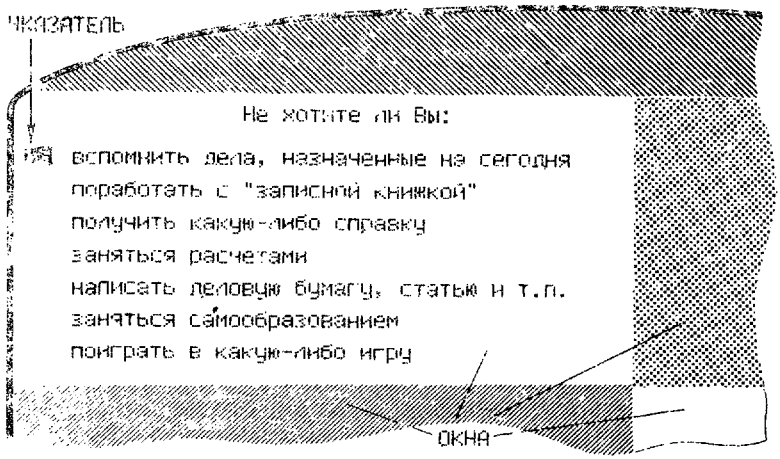


Рис. 12.3. Часть экрана дисплея с меню

Далее было замечено, что такие микроЭВМ (их стали называть персональными ЭВМ) можно превратить: в игральный автомат; устройство подготовки, хранения и редактирования текстов, выводимых на электрическую пишущую машинку; синтезатор музыки, воспроизводимой на динамике, встроенном в клавиатуру или расположенном в телевизоре; таймер и другие устройства, которые могли найти массовый спрос. Для этого надо было в значительной мере упростить способы общения человека с персональной ЭВМ, чтобы сделать ее доступной для любого неквалифицированного (с позиции вычислительной техники) пользователя и даже трехлетнего ребенка. Стремление расширить сферу пользователей ЭВМ привело к созданию новых концепций в разработке программного обеспечения для персональных ЭВМ.

Если первые операционные системы ориентировались на оптимизацию использования аппаратных средств ЭВМ (считающихся наиболее дорогим ресурсом), а затем еще и на поддержание режима коллективного доступа к ЭВМ через сеть дисплеев, то операционные системы персональных ЭВМ ориентированы главным образом на обслуживание индивидуального пользователя и предоставления ему наиболее благоприятного режима общения с ЭВМ. Такие «дружелюбные» операционные системы ведут диалог с пользователем, применяя для этого меню и таблицы разного типа, средства машинной графики и звуковые сигналы, специальные манипуляторы для указания и выборки объектов на экране, «многооконные интерфейсы» и т. п.

Так, после включения персональной ЭВМ на ее экране может появиться текст (рис. 12.3). Передвинув указатель на нужную строку, пользователь инициирует переход к следующему меню, которое заменяет первоначальное или высвечивается рядом (в своем окне). Передвижение указателя и выдачу приказа на выполнение указанного действия можно осуществлять с помощью каких-либо клавиш клавиатуры персональной ЭВМ (тогда в верхней или нижней части экрана создают окно, в котором дается описание этих клавиш). Для тех же целей можно использовать специальное устройство перемещения указателя и отметки нужного его положения: рукоятку с кнопкой («джойстик») или коробочку с кнопкой («мышь»). Поворот такой ручки или перемещение коробочки по столу изменяет положение вмонтированных в них переменных резисторов, что и приводит к изменению положения указателя на экране дисплея.

Средства общения с ЭВМ (человеко-машинный интерфейс) все время улучшаются. На вооружение берутся графика, цвет, звук. Это во многих случаях избавляет пользователя от работы с клавиатурой, т. е. существенно облегчает общение с ЭВМ людям, желающим сыграть с ней в шахматы, домашним хозяйкам (для получения кулинарного рецепта) или ребенку, который хочет порисовать или заняться самообучением.

Владельцам персональных ЭВМ, которые работают над созданием тех или иных текстов (книг, статей, писем, деловых бумаг и т. п.), предоставляются различные программы, позволяющие набирать текст на экране дисплея, редактировать его, выводить на печать и (или) сохранять на дискете или магнитной ленте (см. параграф 7.4). Лучшие из таких программ (которые обычно называют «редактор текста») берут на себя построчную организацию создаваемого текста:

производят грамматически правильный перенос той части слова, которая не размещается в конце строки заданной длины;

выравнивают правый край текста путем увеличения междусловных промежутков.

Кроме того, редакторы текста позволяют производить на экране изменение любой части текста (удаление, замену или вставку буквы, слова или последовательности слов) с автоматической перекомпоновкой строк, расположенных за исправляемым местом текста. Исправлять можно как вновь набираемый текст, так и текст, который хранится на дискете или магнитной ленте. Для этого файл с текстом считывается с магнитного носителя информации в память ЭВМ, на экран дисплея выводится то место текста, в которое надо внести коррективы; текст исправляется и записывается на старое или новое место магнитного носителя (в последнем случае будут сохраняться две версии текста) Такое редактирование текста и (или) распечатку новой его версии можно делать сколь угодно часто. При этом, например, получение «дополнительных» экземпляров статьи сводится к пересылке ее текста с магнитного носителя на печатающее устройство, т. е. не требуется повторения набора и редактирования статьи.

Многие редакторы текста могут работать в режиме, называемом «режь и клей», т. е. позволяют выбирать куски из одного или нескольких документов (статей) и направлять их в буфер «склейки» — область памяти, в которой хранится последний кусок «отрезанного» или «скопированного» текста. Место для вклейки отрезанного куска в тот же (или, если это необходимо, в другой) документ отыскивается с помощью клавиш управления курсором дисплея, после чего нажимается специальная клавиша («клей»). Полный текст из буфера «склейки» вводится в выбранное место требуемого документа.

Если в персональной ЭВМ предусмотрено печатающее устройство, синтезирующее символы из отдельных точек (см. параграф 7.2), то такие устройства часто снабжают контроллерами для обеспечения вывода любых комбинаций точек. В подобных случаях целесообразно использовать такие редакторы текста, которые позволяют работать со смешанной (текст — изображение) информацией. Эти редакторы предоставляют возможность печатать текст различными шрифтами, различающимися по величине, наклону и (или) жирности, а также включают в текст графики, диаграммы, рисунки и т. п.

Графические редакторы, как правило, работают с библиотеками различных законченных изображений и заготовок. В каждую из библиотек включены сотни изображений, относящихся к той или иной профессии или научной дисциплине. Графический редактор обеспечивает самые разнообразные способы слияния изображений или их частей и дополнения получаемых рисунков с помощью простых и удобных средств, работающих на основе

меню и функциональных клавиш. Для того чтобы провести линию, пользователю достаточно переместить курсор в требуемом направлении или же указать начальную и конечную точки отрезка, а остальную работу выполнит программа. Цвет линии задается путем нажатия той или иной функциональной клавиши. Процесс синтеза изображений упрощается при подключении светового пера.

Возможность быстрой смены одного изображения (или его части) на другое, отличающееся от первого лишь некоторыми деталями, позволяет имитировать эффект движения, т. е. «оживлять» изображения. Это свойство широко используется в обучающих и игровых программах, при моделировании тех или иных явлений природы, создании мультфильмов и т. п.

Персональные ЭВМ снабжают хорошими библиотеками программ для решения задач по самым разнообразным отраслям науки и техники. Такие программы обычно работают в диалоговом режиме (с подсказками), и пользователю не нужно иметь специальной подготовки для работы с этими программами. Создано также много обучающих программ по различным предметам начальной, средней и высшей школ, обучению машинописи, правилам уличного движения и т. п. Широко используются программы, обеспечивающие организацию досуга и игр (имеются более тысячи игровых программ).

Для создания новых программ и корректировки существующих (обучающих, игровых и т. п.) все персональные ЭВМ снабжают компиляторами и интерпретаторами с различных языков кодирования программ. На сегодня наиболее популярным стал язык БЕЙСИК, интерпретатор с которого встраивается (записывается в ПЗУ) практически всех персональных ЭВМ. Расширенные версии этого языка облегчают работу с текстами, графикой и музыкальным сопровождением (через громкоговоритель, встроенный в персональную ЭВМ). Наряду с БЕЙСИКом широко используются языки ПАСКАЛЬ, ФОРТРАН, С.

Глава 13

ТЕНДЕНЦИИ РАЗВИТИЯ ЭВМ

13.1. Поколения ЭВМ и эпохальные события в развитии вычислительной техники

История развития вычислительной техники насчитывает всего около 40 лет. Однако за это время уже сменилось четыре поколения ЭВМ, существенно отличающихся друг от друга: элементной базой, методами разработки программного обеспечения, конструкторско-технологическим исполнением, степенью унификации технических и программных средств, перечнем предоставляемых услуг, удобством общения пользователей с ЭВМ и др. Причем именно элементная база ЭВМ наиболее точно характеризует различные поколения ЭВМ.

Первое поколение ЭВМ — это машины, построенные на электронных лампах. Быстродействие ЭВМ этого поколения не превышало 10—20 тыс. операций в секунду. Они были больших размеров, не отличались высокой надежностью, нуждались в громоздких системах охлаждения.

ЭВМ второго поколения были построены на полупроводниковых приборах — транзисторах, они стали более надежными, их быстродействие возросло до сотен тысяч операций в секунду.

В ЭВМ третьего поколения в качестве элементной базы используются в основном интегральные схемы, что обеспечило построение ЭВМ высокой надежности и с быстродействием порядка миллионов операций в секунду.

ЭВМ четвертого поколения — это многомашинные и многопроцессорные вычислительные комплексы, построенные на основе больших интегральных схем. ЭВМ этого поколения отличает высокое быстродействие (десятки миллионов операций в секунду) и высокая степень надежности.

Развитие элементной базы и соответствующая смена поколений ЭВМ отражают в основном только технические и технологические аспекты развития вычислительной техники. Вместе с тем в истории развития средств вычислительной техники можно выделить несколько моментов, существенным образом определяющих значимость вычислительной техники для общественного прогресса. По мнению акад. Н. Н. Моисеева, следует выделять следующие события, имеющие эпохальное значение:

изобретение в конце 40-х годов ЭВМ;

создание в конце 60-х годов третьего поколения ЭВМ;

создание в 70-х годах микропроцессоров и сетей ЭВМ;

появление на рубеже 80-х годов персональных ЭВМ.

13.2. Классификация ЭВМ и тенденции их развития

Все многообразие существующих в настоящее время ЭВМ можно разделить на четыре основных класса:

1) микроЭВМ, в том числе и персональные ЭВМ;
2) мини-ЭВМ;
3) ЭВМ общего назначения;
4) суперЭВМ.

Такая классификация ЭВМ позволяет учесть как конструкторско-технологические особенности ЭВМ различных моделей, так и возможные области их эффективного применения. Рассмотрим характерные особенности и тенденции развития отдельных классов ЭВМ, не останавливаясь на микроЭВМ, поскольку им была посвящена гл. 11.

Мини-ЭВМ — это малые вычислительные машины, ориентированные на широкий круг научных и инженерных применений: автоматизированные системы научных исследований; автоматизированные рабочие места проектировщиков, конструкторов, технологов; управление технологическим оборудованием, технологическими линиями и участками (ГПС, ИПК); системы управления в связи и на транспорте; инженерные и научные расчеты и др. Как видно из приведенного перечня, мини-ЭВМ предназначены для использования в конкретной предметной области, и именно на это ориентируются разработчики мини-ЭВМ, определяя их технические характеристики: быстродействие, емкость памяти, номенклатуру внешних устройств, состав программного обеспечения и т. п. Мини-ЭВМ отличают невысокая стоимость (десятки тысяч рублей), простота обслуживания и высокая надежность. Для размещения мини-ЭВМ не требуется больших помещений (например, не более 20 м^2 для ЭВМ типа СМ4).

К мини-ЭВМ относятся ряд таких популярных машин, как СМ1, СМ2, СМ3, СМ4, СМ1420, и различные измерительно-вычислительные комплексы (ИВК), созданные на их основе.

ЭВМ общего назначения (ЕС ЭВМ) представляют собой семейство машин, в которое входят ЭВМ различной производительности, оснащенные широким набором внешних устройств, построенные на единых конструкторско-технологических принципах, обладающие программной совместимостью. ЭВМ этого класса предназначаются для решения широкого круга

задач: научно-технических, планово-экономических, учетно-статистических, информационно-поисковых, а также для работы в рамках САПР, АСУП, АСТПП и т. п.

ЭВМ единой серии (ЕС ЭВМ) разрабатываются совместно странами — членами СЭВ с 1969 г. За это время были реализованы три международных проекта по разработке семейств ЭВМ: ЕС ЭВМ-1, ЕС ЭВМ-2 и ЕС ЭВМ-3. По этим проектам за 16 лет совместной работы созданы [7, 13]: 32 модели ЭВМ; семь вычислительных комплексов; более 180 моделей различных внешних устройств.

В современных науке и производстве есть целый ряд задач, решение которых либо экономически невыгодно, либо вообще невозможно с использованием даже ЭВМ общего назначения — их производительности недостаточно. Для решения такого рода задач создаются ЭВМ рекордной производительности — суперЭВМ. Как правило, это многопроцессорные вычислительные комплексы с суммарным быстродействием десятки или даже сотни миллионов операций в секунду. Отечественная промышленность выпускает несколько моделей суперЭВМ: «Эльбрус-1» и «Эльбрус-2», ПС-2500, ПС-3000, старшие модели семейства ЕС ЭВМ-3.

Укажем здесь только предметные области, решение задач в которых требует использования суперЭВМ.

1. Автоматизация проектирования сложных объектов и систем, таких как летательные аппараты, сверхбольшие интегральные схемы и т. п.

2. Управление космическими объектами.

3. Обработка больших объемов информации, получаемых со спутников и при аэрофотосъемке.

4. Задачи вычислительной аэродинамики, физики элементарных частиц, сейсмологии и др.

Завершая разговор о классификации современных ЭВМ, укажем основные тенденции развития вычислительной техники на ближайшие 10—15 лет.

1. Интеграция ЭВМ в крупные информационно-вычислительные системы (локальные и глобальные сети ЭВМ), а также рассредоточение и приближение средств вычислительной техники к источникам информации (встраивание микропроцессоров и микроЭВМ в приборы, системы, технологическое оборудование и т. п.).

2. Разработка и внедрение унифицированных модульных архитектур ЭВМ, обеспечивающих возможность поэтапного расширения и наращивания вычислительной мощности ЭВМ без необходимости изменения устройств памяти, периферийного оборудования и даже программного обеспечения.

В целях развития ЭВМ с унифицированными модульными архитектурами требуется разработка:

эффективных методов и средств параллельных и распределенных вычислений;

микропроцессоров большой вычислительной мощности, сопоставимой с процессорами современных ЭВМ;

СБИС памяти, равных по емкости системам памяти современных ЭВМ.

3. Интегральная полупроводниковая схемотехника останется основой элементной базы всех моделей ЭВМ в ближайшие годы.

4. Появление нового поколения ЭВМ обычно связывалось с созданием новой элементной базы. Теперь очередной скачок в развитии вычислительной техники произойдет не столько в области элементной базы ЭВМ, сколько в разработке новых конфигураций ЭВМ, центральных и специализированных процессоров, средств комплексирования ЭВМ и др.

Широкое внедрение ЭВМ во все сферы человеческой деятельности потребует решения еще целого ряда масштабных задач. Все ведущие страны прилагают значительные усилия в области развития вычислительной техники, что отражено в проектах вычислительных систем пятого поколения.

13.3. На пути к вычислительным системам пятого поколения

Задача разработки принципиально новых компьютеров в достаточно сжатые сроки впервые была поставлена в 1979 г. японскими специалистами. В рамках решения этой задачи в Японии находится в стадии реализации ряд проектов развития вычислительной техники, обеспечивающих, в частности, создание к 1990 г. сверхбыстродействующей ЭВМ (с производительностью в 100 раз выше, чем у современных суперЭВМ), а к 1991 г.— первых вычислительных систем пятого поколения (ВСПП) [36].

Какими же характерными особенностями должны обладать ВСПП?

1. Развитый человеко-машинный интерфейс, в том числе использование для общения с ЭВМ естественных языков, устройств распознавания речи и изображений.

2. Работа с базами знаний в конкретных предметных областях и организация на их основе систем искусственного интеллекта.

Такого рода возможности ВСПП могут быть достигнуты только благодаря решению следующих научных и производственных проблем:

1) разработка теоретических основ построения ЭВМ с искусственным интеллектом;

2) разработка эффективных языков логического программирования (первыми их образцами можно считать ПРОЛОГ и ЛИСП);

3) разработка и создание на основе передовых технологий СБИС, содержащих от 1 до 10 млн логических элементов на кристалле;

4) разработка и реализация новых архитектур ЭВМ и вычислительных комплексов, обеспечивающих параллельную и распределенную обработку информации.

Реализация поставленных задач позволит создать ЭВМ нового (пятого) поколения, обладающие быстродействием до 10^{11}—10^{12} опер./с и объемами оперативной памяти до 10^{12} байт. Вместе с тем производительность ЭВМ будет оцениваться количеством логических выводов в единицу времени и для ЭВМ пятого поколения достигнет 10^6—10^9 логических выводов в секунду.

Даже при современном высоком уровне развития науки и техники проекты создания ВСПП могут показаться фантастическими. Но развитые страны приняли вызов Японии и интенсивно работают в этом направлении:

ряд фирм США объединил свои усилия в области научных исследований в области ВСПП (проект «α/Ω»), план работ рассчитан на 6—10 лет, причем за первые пять лет планируется затратить около 700 млн долл.;

в Англии реализуется «проект Альви», направленный на создание ВСПП, стоимость проекта 550 млн долл. на пять лет;

страны Европейского экономического сообщества (ФРГ, Франция и др.) объявили о разработке проекта ВСПП под названием Esprit.

Научные исследования в области создания ВСПП ведутся и в СССР.

Завершая разговор о ВСПП, следует отметить, что их создание и всемерное внедрение в научную и производственную практику потребуют концентрации усилий не только отдельных организаций и фирм, но и целых государств. Именно настолько масштабна, приоритетна и необходима для общественного прогресса в целом проблема создания ВСПП.

ПРИЛОЖЕНИЯ

Приложение А

Таблица перевода чисел из одной системы счисления в другую

Таблица А.1

Сравнение десятичных, двоичных, восьмеричных и шестнадцатеричных кодов чисел

Десятичные	Двоичные	Восьмеричные	Шестнадцатеричные	Десятичные	Двоичные	Восьмеричные	Шестнадцатеричные	Десятичные	Двоичные	Восьмеричные	Шестнадцатеричные
0	0	0	0	11	1011	13	B	22	1 0110	26	16
1	1	1	1	12	1100	14	C	23	1 0111	27	17
2	10	2	2	13	1101	15	D	24	1 1000	30	18
3	11	3	3	14	1110	16	E	25	1 1001	31	19
4	100	4	4	15	1111	17	F	26	1 1010	32	1A
5	101	5	5	16	1 0000	20	10	27	1 1011	33	1B
6	110	6	6	17	1 0001	21	11	28	1 1100	34	1C
7	111	7	7	18	1 0010	22	12	29	1 1101	35	1D
8	1000	10	8	19	1 0011	23	13	30	1 1110	36	1E
9	1001	11	9	20	1 0100	24	14	31	1 1111	37	1F
10	1010	12	A	21	1 0101	25	15	32	10 000	40	20

Таблица А.2

Значения степеней числа 2

n	2^n	n	2^n	n	2^n	n	2^n	n	2^n
0	1	4	16	8	256	12	4 096	16	65 536
1	2	5	32	9	512	13	8 192	17	131 072
2	4	6	64	10	1 024	14	16 384	18	262 144
3	8	7	128	11	2 048	15	32 768	19	524 288

Таблица А.3

Преобразование десятичных чисел в шестнадцатеричные числа

Множитель	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
10	A	14	1E	28	32	3C	46	50	5A
10^2	64	C8	12C	190	1F4	258	2BC	320	384
10^3	3E8	7D0	BB8	FA0	1388	1770	1B58	1F40	2328
10^4	2710	4E20	7530	9C40	C350	EA60	11170	13880	15F90
10^5	186A0	30D40	493E0	61A80	7A120	927C0	AAE60	C3500	DBBA0
10^6	F4240	1E8480	2DC6C0	3D0900	4C4B40	5B8D80	6ACFC0	7A1200	895440

Пример: $1234 = 1000 + 200 + 30 + 4 = (3E8)_{16} + (C8)_{16} + (1E)_{16} + (4)_{16} = (4D2)_{16}$

Таблица А.4

Преобразование шестнадцатеричных чисел в десятичные числа

Множитель	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
16	16	32	48	64	80	96	112
16^2	256	512	768	1024	1280	1536	1792
16^3	4096	8192	12288	16384	20480	24576	28672
16^4	65536	131072	196608	262144	327680	393216	458752
16^5	1048576	2097152	3145728	4194304	5242880	6291456	7340032

Множитель	8	9	A	B	C	D	E	F
1	8	9	10	11	12	13	14	15
16	128	144	160	176	192	208	224	240
16^2	2048	2304	2560	2816	3072	3328	3584	3840
16^3	32768	36864	40960	45056	49152	53248	57344	61440
16^4	524288	589824	655360	720896	786432	851968	917504	983040
16^5	8388608	9437184	10485760	11534336	12582912	13631488	14680064	15728640

Пример: $(FDEC)_{16} = (F)_{16} \cdot 16^3 + (D)_{16} \cdot 16^2 + (E)_{16} \cdot 16 + (C)_{16} =$
$= 61440 + 3584 + 208 + 12 = 65244$

Приложение Б

Таблицы кодирования информации, передаваемой между отдельными составляющими вычислительных систем

Таблица Б.1

Кодирование символов 8-разрядными (байтовыми) кодами, используемыми для обмена информацией (КОИ-8)

Младшие цифры	Старшие цифры 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ПУС	АР1		0	@	P	`	p	ВЦФ				ю	п	Ю	П
1	НЗ	СУ1	!	1	A	Q	a	q	НЗН				а	я	А	Я
2	НТ	СУ2	"	2	B	R	b	r	РП	УУК			б	р	Б	Р
3	КТ	СУ3	#	3	C	S	c	s					ц	с	Ц	С
4	КП	СТП	¤	4	D	T	d	t	БК	ВКП			д	т	Д	Т
5	КТМ	НЕТ	%	5	E	U	e	u	НС	ОСУ			е	у	Е	У
6	ДА	СИН	&	6	F	V	f	v	НП	ВП			ф	ж	Ф	Ж
7	ЗВ	КБ	'	7	G	W	g	w	ОЖД				г	в	Г	В
8	ВШ	АН	(	8	H	X	h	x					х	ь	Х	Ь
9	ГТ	КН	)	9	I	Y	i	y					и	ы	И	Ы
A	ПС	ЗН	*	:	J	Z	j	z	УР				й	з	Й	З
B	ВТ	АР2	+	;	K	[	k	{	СП2	СП3			к	ш	К	Ш
C	ПФ	РФ	,	<	L	\	l	|		ВЫП			л	э	Л	Э
D	ВК	РГ	-	=	M	]	m	}		ВСТ			м	щ	М	Щ
E	Вых	РЗ	.	>	N	¬	n	‾	НРВ				н	ч	Н	Ч
F	Вх	РЗ	/	?	O	_	o	ЗБ	СП1				о	ъ	О	ЗБ

Таблица Б.2

Стандартные управляющие сигналы

Код	Обозначение: Русское	Обозначение: Английское	Наименование символа	Код	Обозначение: Русское	Обозначение: Английское	Наименование символа
00	ПУС	NUL	Пусто	19	КН	EM	Конец носителя
01	НЗ	SOH	Начало заголовка	1A	ЗМ	SUB	Замена
02	НТ	STX	Начало текста	1B	АР2	ESC	Авторегистр 2
03	КТ	ETX	Конец текста	1C	РФ	FS	Разделитель файлов
04	КП	EOT	Конец передачи	1D	РГ	GS	Разделитель групп
05	КТМ	ENQ	Кто там ?	1E	РЗ	RS	Разделитель записей
06	ДА	ACK	Подтверждение	1F	РЭ	US	Разделитель элементов
07	ЗВ	BEL	Звонок	80	ВЦФ	DS	Выбор цифры
08	ВШ	BS	Возврат на шаг	81	НЗН	SOS	Начало значимости
09	ГТ	HT	Горизонтальная табуляция	82	РП	FS	Разделитель полей
				84	БК	BYP	Блокировка
0A	ПС	LF	Перевод строки	85	НС	NL	Новая строка
0B	ВТ	VT	Вертикальная табуляция	86	НП	LC	Нижнее положение
0C	ПФ	FF	Перевод формата	87	ОЖД	IL	Ожидание
0D	ВК	CR	Возврат каретки	8A	УР	SM	Установление режима
0E	ВЫХ	SO	Выход	8B	СП2	CU2	Символ пользователя 2
0F	ВХ	SI	Вход	8E	НРВ	SMM	Начало ручного ввода
10	АР1	DLE	Авторегистр 1	8F	СП1	CU1	Символ пользователя 1
11	СУ1	DC1	Символ управления 1	92	УУК	CC	Управление указателем
12	СУ2	DC2	Символ управления 2	94	ВКП	PN	Включение перфоратора
13	СУ3	DC3	Символ управления 3	95	ОСУ	RS	Остановка считывающего устройства
14	СТП	STOP	Стоп				
15	НЕТ	NAK	Отрицание	96	ВП	UC	Верхнее положение
16	СИН	SYN	Синхронизация	98	СП3	CU3	Символ пользователя 3
17	КБ	ETB	Конец блоков	9C	ВЫП	PF	Выключение перфоратроа
18	АН	CAN	Аннулирование	9D	ВСТ	RES	Восстановление

Приложение В

Система команд микроЭВМ «Электроника-60»

Ресурсы, используемые командами. На рис. В.1 приведены программно-доступные регистры микроЭВМ «Электроника-60».

Большинство регистров общего назначения (РОН) являются универсальными: выполняют функции аккумуляторов, базовых регистров, указателей адресов и т. п. Два РОНа используются специальным образом: регистр R6 интерпретируется командами работы с подпрограммами и прерываний как

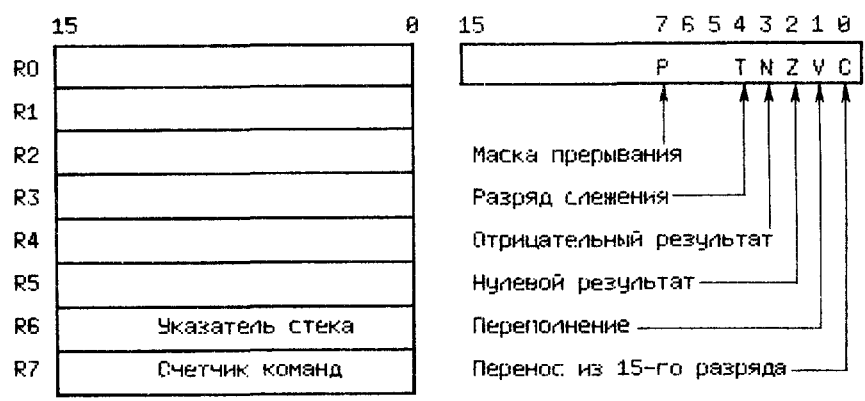


Рис. В.1. Регистры, используемые при программировании

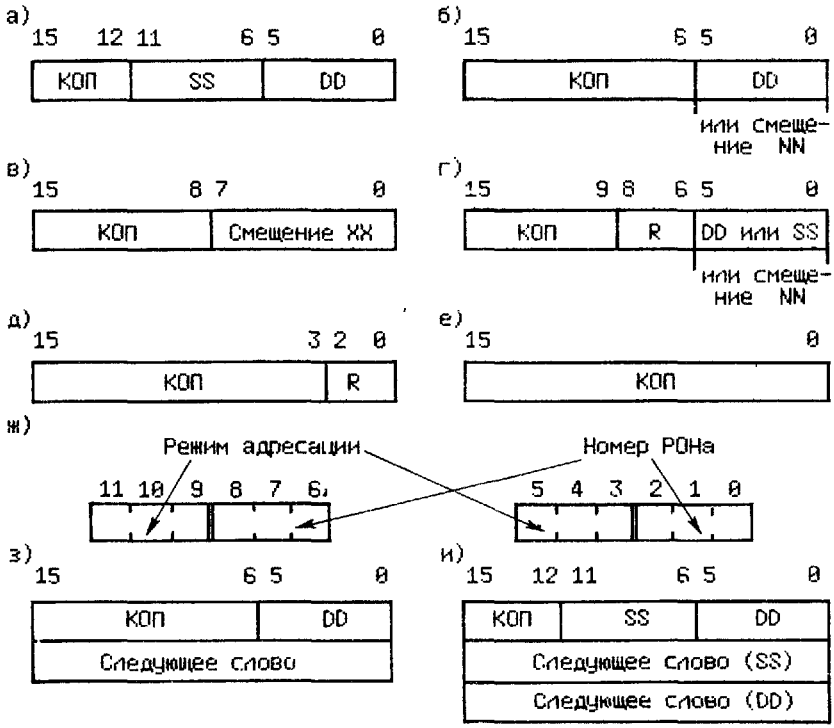


Рис. В.2. Основные форматы команд микроЭВМ «Электроника-60»:

а — двухадресные команды; *б* — одноадресные команды; *в* — команды перехода и специальных прерываний; *г* — арифметические операции с фиксированной точкой и команды JSR, XOR; *д* — команда RTS; *е* — безадресные команды; *ж* — указание операнда в командах формата *а*, *б* и *г*; *з* — возможные форматы одно- и двухадресных команд

аппаратный указатель стека, а регистр R7 служит счетчиком команд и не может выполнять функций аккумулятора.

В регистре состояний хранятся коды условий (N, Z, V, C), информирующие о результате выполнения операций, маска прерывания (P) и разряд слежения (T). Установкой T=1 программист обеспечивает возможность покомандного выполнения программ при их отладке.

Форматы команд. В микроЭВМ «Электроника-60» используются шесть основных форматов команд (рис. В.2), коды операций (КОП) которых имеют различную длину. В зависимости от режима адресации команды с форматами рис. В. 2, *б, г* могут использовать дополнительное слово памяти (т. е. стать 32-разрядными), а команды с форматами рис. В. 2, *а* — даже два дополнительных слова памяти (рис. В.2, *з, и*).

Вся адресация производится только через РОНы. В адресной части (частях) команды указывается либо номер РОНа (*R*), либо шестибитовое адресное поле (рис. В. 2, *ж*), которое определяет источник (SS) и (или) приемник (DD) информации. Только в случае применения при адресации регистра R7 (счетчика команд) дополнительные слова команды используются машиной в качестве операнда (см. рис. В. 5, *а*) или адреса операнда (см. рис. В. 5. *б*). В остальных случаях используется лишь косвенное адресование (через РОНы) к словам (байтам) памяти микроЭВМ.

Восьмибитовые поля (XX) команд переходов и шестибитовые поля (NN) команд MARK и SOB (форматы рис. В. 2. *б, г*) служат для хранения смещения, используемого при определении адреса перехода.

Список команд. Основные команды микроЭВМ «Электроника-60» приведены в табл. В. 1. — В. 3. Следует остановиться на следующих их особенностях.

1. В командах, которые могут оперировать со словами и байтами, приводятся два мнемонических обозначения (байтовые команды отмечаются

Таблица В.1

Список двух- и одноадресных команд микроЭВМ «Электроника-60»

Мнемоника	Восьмеричный код	Формат	Название и описание	Коды условий после выполнения команды
			Группа двухадресных команд	
MOV MOVB	01SSDD 11SSDD	a	**Пересылка.** Содержимое SS пересылается в DD	N=1, если (SS)<0 Z=1, если (SS)=0 V=0; C не изменяется
CMP CMPB	02SSDD 12SSDD	a	**Сравнение.** Установка кодов условий в соответствии с результатом вычитания содержимого DD из содержимого SS. Результат вычитания никуда не записывается	N=1, если RES<0 Z=1, если RES=0 V=1, если возникло арифметическое переполнение C=1, если был заем, т. е. не было переноса из старшего разряда
BIT BITB	03SSDD 13SSDD	a	**Проверка битов.** Установка кодов условий в соответствии с результатом логической операции И над содержимыми SS и DD (без записи результата)	N=1, если старший разряд RES равен 1 Z=1, если RES=0 V=0; C не изменяется
BIC BICB	04SSDD 14SSDD	a	**Очистка битов.** Обращает в 0 те разряды (DD), которым соответствуют установленные в 1 разряды (SS)	N=1, если старший разряд RES равен 1 Z=1, если RES=0 V=0, C не изменяется
BIS BISB	05SSDD 15SSDD	a	**Установка битов (ИЛИ).** Установка в 1 тех разрядов (DD), которым соответствуют установленные в 1 разряды (SS)	N=1, если старший разряд RES=1 Z=1, если RES=0 V=0; C не изменяется
ADD	06SSDD	a	**Сложение.** Содержимое SS складывается с содержимым DD и результат пересылается в DD	N=1, если RES<0 Z=1, если RES=0 V=1, если возникло арифметическое переполнение C=1, если был перенос из старшего разряда результата
SUB	16SSDD	a	**Вычитание.** Содержимое SS вычитается из содержимого DD и результат пересылается в DD	N=1, если RES<0 Z=1, если RES=0 V=1, если возникло арифметическое переполнение C=1, если был заем, т. е. не было переноса из старшего разряда результата
MUL	070RSS	г	**Умножение.** Содержимое SS умножается на содержимое R и результат (два слова) пересылается в R и следующий за ним регистр	N=1, если RES<0 Z=1, если RES=0 V=0 C=1, если RES<-2^{15} или RES>$2^{15}-1$

Мнемоника	Восьмеричный код	Формат	Название и описание	Коды условий после выполнения команды
DIV	071RSS	г	**Деление.** Содержимое R и следующего за ним регистра делится на содержимое SS. Частное помещается в R, а остаток — в следующий за ним регистр	N=1, если RES<0 Z=1, если RES=0 V=1, если (SS)=0 или абсолютное значение делимого больше абсолютного значения делителя C=1, если (SS)=0
XOR	074RSS	г	**Исключающее ИЛИ.** Над содержимым R и SS выполняется операция XOR (см. п. 6)	N=1, если RES<0 Z=1, если RES=0 V=0; C не изменяется
			Группа одноадресных команд	
CLR CLRB	0050DD 1050DD	б	**Очистка.** Обращает в 0 содержимое DD	N=V=C=0 Z=1
COM COMB	0051DD 1051DD	б	**Инвертирование.** Получение обратного кода (инверсии) содержимого DD (результат пересылается в DD)	N=1, если старший разряд RES равен 1 Z=1, если RES=0 V=0; C=1
INC INCB	0052DD 1052DD	б	**Увеличение.** Содержимое DD увеличивается на 1 (результат пересылается в DD)	N=1, если RES<0 Z=1, если RES=0 V=1, если (DD)=77777 или 177 (при работе с байтами) C не изменяется
DEC DECB	0053DD 1053DD	б	**Уменьшение.** Содержимое DD уменьшается на 1 (результат пересылается в DD)	N=1, если RES<0 Z=1, если RES=0 V=1, если (DD)=100000 или 200 (при работе с байтами) C не изменяется
NEG NEGB	0054DD 1054DD	б	**Изменение знака.** Содержимому DD присваивается знак минус (содержимое DD заменяется на свое двоичное дополнение)	N=1, если RES<0 Z=1, если RES=0 V=1, если (DD)=100000 или 200 (при работе с байтами) C=0, если RES=0
ADC ADCB	0055DD 1055DD	б	**Сложение с переносом.** Содержимое разряда C регистра состояний складывается с содержимым DD (результат пересылается в DD)	N=1, если RES<0 Z=1, если RES=0 V=1, если C было равно 1 а (DD)=077777 или 177 (при работе с байтами) C=1, если C было равно 1, а (DD)=177777 или 377 (при работе с байтами)
SBC SBCB	0056DD 1056DD	б	**Вычитание переноса.** Содержимое разряда C регистра состояний вычитается из содержимого DD (результат пересылается в DD)	N=1, если RES<0 Z=1, если RES=0 V=1, если C было равно 1, а (DD)=100000 или 200 (при работе с байтами) C=1, если C было равно 1, а (DD)=0
TST TSTB	0057DD 1057DD	б	**Проверка.** Установка кодов условий в соответствии с содержимым DD	N=1, если RES<0 Z=1, если RES=0 V=C=0

Мнемоника	Восьмеричный код	Формат	Название и описание	Коды условий после выполнения команды
ROR RORB	0060DD 1060DD	б	Циклический сдвиг вправо. Все биты содержимого смещаются на 1 разряд вправо. Младший разряд (DD) переходит в C, а C — в старший разряд (DD)	N=1, если старший разряд RES равен 1 Z=1, если RES=0 V=N XOR C C загружается младшим битом (DD)
ROL ROLB	0061DD 1061DD	б	Циклический сдвиг влево. Работает аналогично ROR, но сдвиг влево	N, Z, V — см ROR C загружается старшим битом (DD)
ASR ASRB	0062DD 1062DD	б	Арифметический сдвиг вправо. Все биты (DD), кроме старшего, смещаются на 1 разряд вправо. Младший бит (DD) переходит в C	N=1, если RES<0 Z=1, если RES=0 V=N XOR C C загружается младшим разрядом (DD)
ASL ASLB	0063DD 1063DD	б	Арифметический сдвиг влево. Все биты (DD), кроме старшего, смещаются на 1 разряд влево. В младший разряд (DD) записывается 0, а C заполняется старшим разрядом (DD)	N=1, если RES<0 V=N XOR C C загружается старшим разрядом (DD)
SWAB	0003DD	б	Обмен (свопинг) байтов. В операнде (DD) — перестановка байтов. Результат оценивается по младшему байту	N=1, если 7-й разряд (DD) равен 1 Z=1, если младший байт равен 0 V=C=0
SXT	0067DD	б	Расширение знака. Содержимое DD заполняется нулями или единицами в зависимости от значения N (0 или 1)	Z=1, если (DD)=0 N, V и C не изменяются

дополнительным символом B) и соответствующие им восьмеричные коды команд.

2. Использование восьмеричного (а не шестнадцатеричного) кодирования команд связано с тем, что поля, определяющие адреса операндов, кратны трем, а расположенные в старших разрядах 4-, 7-, 13- или 16-битовые коды операций (КОП) легко кодируются соответственно двумя, тремя, пятью или шестью восьмеричными цифрами, первая из которых 0 или 1. Например, команда вычитания числа, содержащегося в регистре 5, из числа, содержащегося в регистре 4, т. е. команда с кодом $(1110000101000100)_2$, записывается (и расшифровывается машиной) в виде $(160504)_8$, а не в шестнадцатеричном коде $(E144)_{16}$.

3. Шестибитовые поля адресации источника (Source) и приемника (Destination) информации кодируются в командах символами SS и DD соответственно, трехбитовые поля регистров — символом R, а смещения — теми же символами NN и XX, что и на рис. B. 2 . При этом надо учитывать, что смещение XX — восьмибитовое и его старшие биты, а также младший бит КОП команд перехода создают одну общую восьмеричную цифру. Поэтому последняя цифра КОП команд перехода при определенном значении смещения может измениться с 0 на 1, 2 или 3 и с 4 на 5, 6 или 7.

4. В описаниях команд для обозначения содержимого регистра или слова памяти, на которые указывают SS и DD, используется запись (SS) и (DD).

Таблица В.2

Список команд управления программой микроЭВМ «Электроника-60»

Мнемоника	Восьмеричный код	Формат	Название и описание
JMP	0001DD	б	**Переход.** Передает управление по адресу, определяемому операндом DD
BR	0004XX	в	**Безусловный переход.** Адрес перехода этой и 14 следующих команд определяется по правилам, изложенным в пп. 9 и 10
BNE	0010XX	в	Переход, если не равно нулю (Z=0)
BEQ	0014XX	в	Переход, если равно нулю (Z=1)
BCE	0020XX	в	Переход, если больше или равно нулю (N XOR V=0)
BLT	0024XX	в	Переход, если меньше нуля (N XOR V=1)
BGT	0030XX	в	Переход, если больше нуля (Z OR (N XOR V) =0)
BLE	0034XX	в	Переход, если меньше или равно нулю (Z OR (N XOR V) =1)
BPL	1000XX	в	Переход по плюсу (N=0)
BMI	1004XX	в	Переход по минусу (N=1)
BHI	1010XX	в	Переход, если больше (S OR Z =0)
BLOS	1014XX	в	Переход, если меньше или равно (C OR Z=1)
BVC	1020XX	в	Переход, если нет переполнения (V=0)
BVS	1024XX	в	Переход, если переполнение (V=1)
BCC	1030XX	в	Переход, если нет переноса (C=0)
BCS	1034XX	в	Переход, если перенос (C=1)
SOB	077RNN	г	**Вычитание и переход.** При каждом выполнении SOB содержимое R уменьшается на 1, и если (R) ≠ 0, то выполняется возврат на NN слов. При (R) = 0 осуществляется следующая за SOB команда программы
JSR	004RDD	г	**Обращение к подпрограмме.** Содержимое R заносится в стек, в R записывается адрес возврата (содержимое счетчика команд) и в счетчик команд заносится DD (адрес первой команды подпрограммы). Подпрограмма должна сохранять значение R, чтобы можно было выполнить возврат в вызывающую программу по команде RTS R Если в качестве R используется счетчик команд (R7), то обеспечивается автоматическое сохранение в стеке адреса возврата из подпрограммы
RTS	00020R	д	**Возврат из подпрограммы.** Содержимое R (адрес возврата) пересылается в счетчик команд и восстанавливается (извлекается из стека) прежнее значение R Если в качестве R используется счетчик команд (R7), то адрес возврата извлекается из стека
MARK	0064NN	б	**Возврат из подпрограммы с восстановлением указателя стека.** По команде MARK NN стек освобождается от NN параметров, которые были занесены в него перед обращением к подпрограмме, и реализуются те же функции, что и при выполнении RTS
RTI	000002	е	**Возврат из подпрограммы обработки прерывания.** Обеспечивает извлечение из стека двух верхних слов (занесенных в стек в момент прерывания программы) и загрузку их в счетчик команд (R7) и регистр состояния процессора

5. Формат команды указан в табл. В. 1 и В. 2 в виде ссылки на соответствующий формат рис. В. 2.

6. Коды условий (N, Z, V, C) устанавливаются по результату выполнения операции (RES), значению одного из операндов и (или) типу операции.

Таблица В.3

Список служебных инструкций микроЭВМ «Электроника-60»

Мнемоника	Восьмеричный код	Формат	Назначение и описание
CLC	000241	е	Очистка C (C=0)
CLV	000242	е	Очистка (V=0)
CLZ	000244	е	Очистка (Z=0)
CLN	000250	е	Очистка (N=0)
SEC	000261	е	Установка C (C=1)
SEV	000262	е	Установка (V=1)
SEZ	000264	е	Установка (Z=1)
SEN	000270	е	Установка (N=1)
CCC	000257	е	Очистка признаков (C=V=Z=N=0)
SCC	000277	е	Установка признаков (C=V=Z=N=1)
NOP	000240	е	Нет операции
HALT	000000	е	Останов работы процессора и прекращение передач по общей шине. Работа программы может быть продолжена нажатием клавиши ПРОДОЛЖЕНИЕ
RESET	000005	е	Сброс ВУ установка ВУ в исходное состояние
WAIT	000001	е	Ожидание прерывания производит приостановку выполнения программы до поступления прерывания. По завершении работы подпрограммы обработки прерываний выполнение программы возобновляется с той команды, которая следует за командой WAIT
MFPS	1067DD	б	Чтение содержимого регистра состояний
MTPS	1064DD	б	Запись содержимого регистра состояний

При этом в командах, которые могут оперировать со словами и байтами, в качестве результата (операнда) берется соответственно слово или один из байтов этого слова. Например, если в пятой графе табл. В. 1 записано: «N=1, если RES<0», то это означает, что N будет установлено в 1, когда будет равен 1 либо 15-й бит результата (при работе со словом или старшим байтом), либо 7-й бит результата (при работе с младшим байтом).

7. В некоторых командах бит V регистра состояний устанавливается по результату выполнения логической операции «Исключающее ИЛИ» (XOR) над разрядами N и C: V=N XOR C. Эта операция, называемая иначе «неравнозначность», дает следующие результаты:

0 XOR 0=0, 0 XOR 1=1, 1 XOR 0=1, 1 XOR 1=0

8. Все числа в табл. В. 1 — В. 2 восьмеричные.

9. При побайтовой организации памяти (см. рис. 5.3, *б*) «Электроника-60» работает с 16-разрядными словами, т. е. после выборки однословных команд или данных содержимое счетчика команд наращивается не на 1, а на 2. При выборке каждого дополнительного слова команды или данных счетчик команд опять увеличивается на 2.

10. Команды условного перехода и команда BR позволяют осуществить передачу управления на расстояние, ограниченное смещением XX, интерпретируемым машиной как целое число со знаком в дополнительном коде: от $(10\,000\,000)_2=-128$ до $(01\,111\,111)_2=+127$. В случае выполнения условия это число удваивается машиной, его 7-й бит копируется в старших разрядах (с 8-го по 15-й) и полученное 16-разрядное число прибавляется к содержимому счетчика команд. Так как в этот момент счетчик команд указывает на команду, следующую за исполняемой, то смещение, равное —1, а не 0, обеспечивает команде переход к самой себе. На рис. В. 3 приведены два примера выполнения команды BR.

11. Команды условного перехода осуществляют или не осуществляют передачу управления после проверки либо отдельных кодов условий, либо комбинаций кодов условий. В последнем случае используются логические

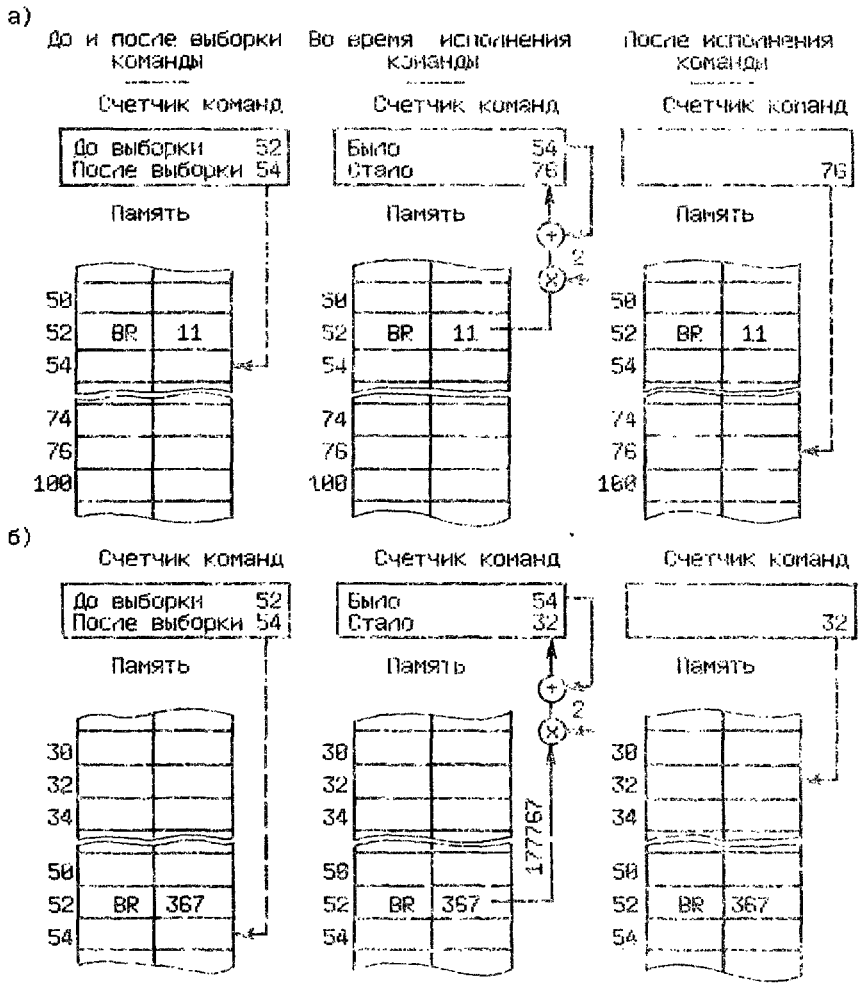


Рис В.3. Выполнение команд передачи управления:
а — команда BR 11; *б* — команда BR 367 (смещение — 11)

операции «Исключающее ИЛИ» (см. п. 7) и «ИЛИ» (OR). Последняя дает следующие результаты:

0 OR 0=0, 0 OR 1=1, 1 OR 0=1 и 1 OR 1=1.

Например, для осуществления перехода в случае, когда результат предшествующих операций оказался меньшим или равным нулю (команда BLE), выясняется, равно ли 1 значение выражения Z OR (N XOR V). При нулевом результате единица содержится в бите Z, при отрицательном же результате — либо в бите N, либо в бите V, так как V=1 только тогда, когда при вычитании происходит переполнение, приводящее к получению неправильного знака (см. параграф 2.3).

12. В табл. В.1 — В.3 отсутствуют команды специальных прерываний (EMT, TRAP, BRT; IOT), возврата из прерываний (RTT) и арифметических сдвигов (ASH и ASHC). С ними можно ознакомиться в работах [13, 22, 41].

Способы адресации операндов. Структура памяти микроЭВМ «Электроника-60» обеспечивает побайтовую адресацию, при этом во многих командах задаются однобайтовые операнды. Большинство команд осуществляет доступ

к двум байтам как к единой структуре данных — слову. Слово имеет четный адрес (расположение байтов в слове показано на рис. 5. 3, *б*). Если команда, для которой требуется операнд длиной в одно слово, пытается осуществить доступ к ячейке с нечетным адресом, то фиксируется ошибка.

Операнд (операнды) в команде задается 6-разрядным полем, включающим 3-разрядный номер регистра (N=0.1, ..., 7) и 3-разрядный указатель режима адресации (рис. В. 2, *ж*). При режиме адресации, равном 0, содержимое регистра используется как операнд; при режимах адресации 1 – 7 содержимое регистра служит для вычисления исполнительного адреса местонахождения операнда в памяти микроЭВМ (рис. В. 4).

0. **Регистровая адресация.** Операндом является содержимое регистра, исполнительный адрес в памяти отсутствует (формат Rn, рис. В. 4, *а*).

1. **Косвенная регистровая адресация.** Исполнительный адрес операнда определяется содержимым регистра (формат @Rn или (Rn), рис. В.4, *б*).

2. **Автоинкрементная адресация.** Исполнительный адрес операнда определяется содержимым регистра, затем содержимое регистра увеличивается на длину операнда в байтах (формат (Rn)+, рис. В. 4, *в*).

3. **Косвенная автоинкрементная адресация.** Косвенный адрес определяется содержимым регистра, затем содержимое регистра увеличивается на 2. Исполнительный адрес выбирается из ячейки памяти по косвенному адресу (формат @ (Rn) + , рис. В. 4, *г*).

4. **Автодекрементная адресация.** Содержимое регистра уменьшается на длину операнда в байтах (1 или 2), затем исполнительный адрес операнда выбирается из регистра (формат — (Rn), рис. В. 4. *д*).

5. **Косвенная автодекрементная адресация.** Содержимое регистра уменьшается на 2, затем косвенный адрес выбирается из регистра, а исполнительный адрес — из ячейки памяти по косвенному адресу (формат @ - (Rn), рис. В. 4, *е*).

6. **Индексация (адресация по базе).** Содержимое регистра (база) и очередного слова в потоке команд (смещение) суммируются для получения исполнительного адреса, а счетчик команд будет указывать на слово, расположенное вслед за смещением (формат E (Rn), рис. В.4, *ж*; E - выражение, определяющее величину смещения). В этом (а также следующем) режиме адресации команда должна иметь одно или два дополнительных слова. Если команда двухадресная и для каждого из операндов задан режим индексации, то в слове, следующем за словом команды, должно размещаться смещение для вычисления адреса первого операнда (SS), а в следующем слове — смещение для вычисления адреса второго операнда (DD), см. рис. В. 2, *з*.

7. **Косвенная индексация.** Содержимое регистра (база) и очередного слова в потоке команд (смещение) суммируется для получения косвенного адреса; исполнительный адрес выбирается из ячейки памяти по косвенному адресу (формат @ E (Rn), рис. В. 4, *з*).

В половине из рассмотренных способов адресации в качестве регистра целесообразно использовать счетчик команд (R7), содержимое которого автоматически увеличивается на 2 после выборки из памяти каждого нового слова потока команд. Это свойство позволяет реализовать дополнительные режимы адресации: непосредственную, абсолютную, относительную и косвенно-относительную.

Непосредственная адресация (режим 2, регистр 7). Операнд содержится в очередном слове команды (формат #E, рис. В. 5, *а*). Содержимое счетчика команд увеличивается на 2, что обусловливает после выборки операнда переход к следующему за ним слову в потоке команд. (Для обеспечения четности показаний счетчика команд его содержимое всегда увеличивается на 2, даже если операндом оказался байт.)

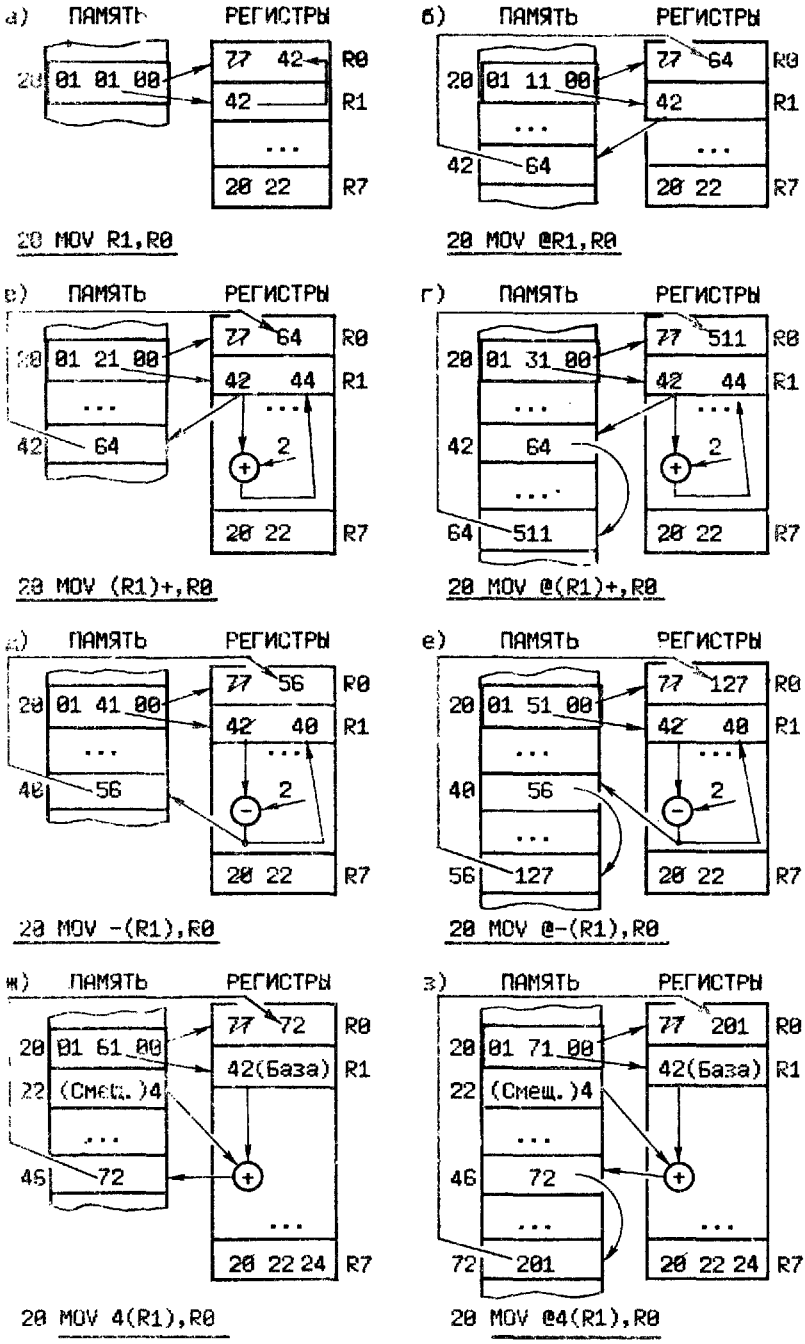


Рис. В.4. Способы адресации:

а, в, д, ж — прямая адресация; *б, г, е, з* — косвенная адресация (Перед пересылкой в R0 и R1 записаны числа 77 и 42 соответственно.)

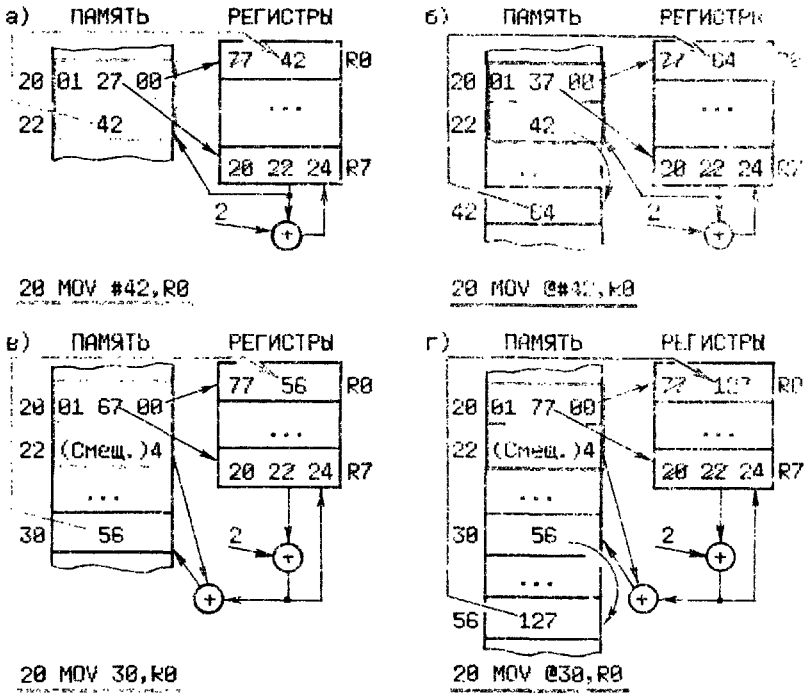


Рис. В.5. Режимы адресации с использованием счетчика команд (R7): *а* — непосредственная адресация, *б* — абсолютная адресация; *в* — относительная адресация; *г* — косвенно-относительная адресация

Абсолютная адресация (режим 3, регистр 7). Исполнительный адрес содержится в очередном слове команды (формат @#E, рис. В. 5, *б*). Содержимое счетчика команд после выборки адреса операнда увеличивается на 2, что обусловливает переход к следующему за ним слову в потоке команд.

Относительная адресация (режим 6, регистр 7). Считывается и временно запоминается очередное слово команды (смещение); содержимое счетчика команд увеличивается на 2, что обусловливает пропуск смещения и переход к следующему за ним слову в потоке команд. Исполнительный адрес получается путем суммирования смещения (сохраняемого слова) с новым содержимым счетчика команд (формат E, рис. В. 5, *в*).

Адресация называется относительной, потому что адрес операнда вычисляется относительно текущего значения счетчика команд: смещение — это расстояние (в байтах) между операндом и словом, на которое указывает счетчик команд в момент вычисления адреса операнда. Если команда и операнд перемещаются в памяти так, что расстояние между ними остается постоянным (равным смещению), то команда будет выполняться правильно в любом месте памяти. Отметим, что мнемоника MOV 30,R0 (рис. В.5, в) с указанием абсолютного адреса операнда не лучшим образом иллюстрирует рассматриваемый способ адресации. Несколько лучше запись MOV 4 (R7), R0, но в этом случае используется формат индексации с регистром R7, а не формат относительной адресации. Обычно используется запись вида MOV A, R0, где A — имя операнда. Ассемблер размещает операнд с именем A в какой-либо ячейке памяти микроЭВМ и при размещении в памяти команды (соответствующей оператору MOV A, R0) определяет расстояние между ними, помещая это расстояние во второе слово команды.

Косвенно-относительная адресация (режим 7, регистр 7). Очередное слово команды (смещение) суммируется с новым содержимым счетчика команд, в результате формируется адрес адреса операнда, т. е. косвенный адрес операнда (формат @E, рис. В. 5, *г*).

Этот вид адресации отличается от предыдущего лишь тем, что формируется не прямой, а косвенный адрес операнда.

Приложение Г

Система команд микропроцессора КР580ИК80А

Ресурсы, используемые командами. На структурной схеме МП (рис. Г. 1) выделены программно-доступные регистры. В основном это однобайтовые регистры, которые могут объединяться в двухбайтовые пары. Регистры и пары регистров кодируются в командах так, как показано в табл. Г. 1.

Следует отметить, что в этой таблице есть кодовые обозначения не только регистров, но и ячеек и памяти (memory), адрес которой (M) находится в регистрах H и L (в H — старшие разряды, в L — младшие). Это сделано потому, что обозначения однобайтовых регистров и M используются в однотипных командах.

Отметим также, что байт признаков в PSW (слово состояния программы — program status word) составляется процессором из однобитовых признаков результата, хранящихся в регистре признаков. Так как таких признаков всего пять, то разряд байта признаков с номером 1 всегда

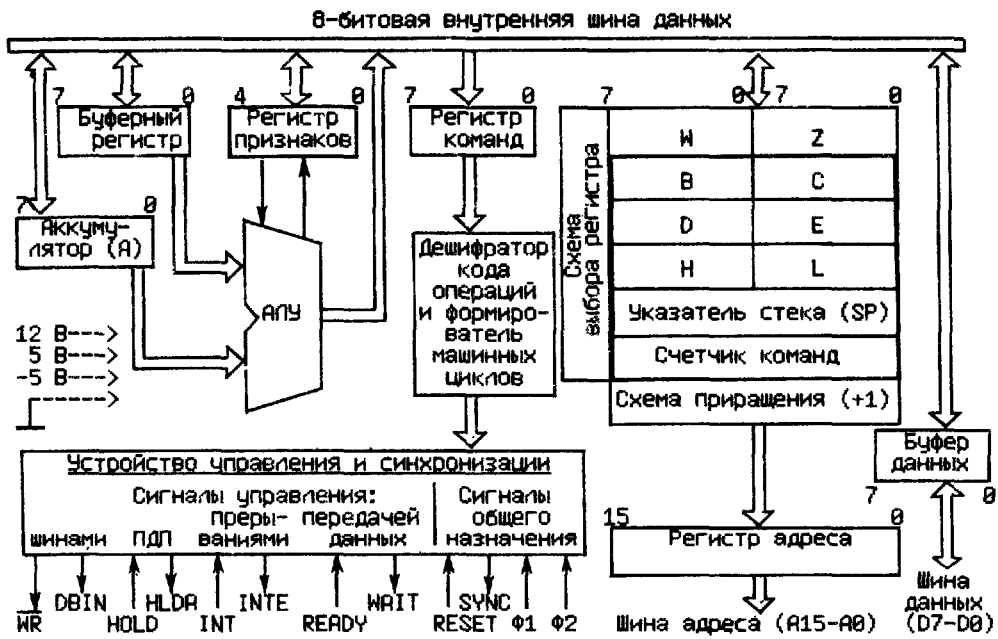


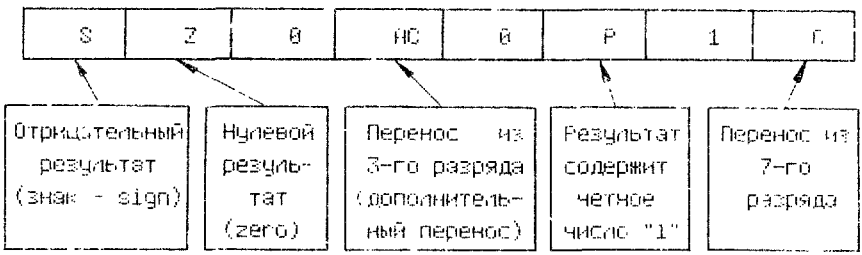
Рис. Г.1. Структура МП КР580ИК80А (выделены программно-управляемые регистры)

Таблица Г.1

Коды регистров и пар регистров, используемые в командах МП

Регистры				Пары регистров			
Код	Имя (r)	Код	Имя (r)	Код (RP)	Имя пары (rp)	Регистры пары	
						старший	младший
000	B	100	H	00	B	B	C
001	C	101	L	01	D	D	E
010	D	110	M (память)	10	H	H	L
011	E	111	A (аккумулятор)	11	PSW	A	Байт признаков

заполняется «1», а разряды с номерами 3 и 5 — «0». Остальные разряды заполняются следующим образом:



Признаки используются в командах условной передачи управления. Коды условий этих команд приведены в табл. Г. 2.

Форматы команд и способы адресации. В МП КР580ИК80А используются 11 форматов команд (рис. Г. 2), коды операций (КОП) которых имеют различную длину (2, 5, 6 или 8 бит) и часто состоят из двух частей. В зависимости от способа адресации команды могут быть одно-, двух- или трехбайтовыми.

В командах используются лишь четыре способа адресации:
регистровая (MOV r1, r2; ADD r; PCHL; ...);
косвенно-регистровая (MOV M, r; ADD M; PUSH; POP; ...);
непосредственная (MVI r, data 8; ADI data 8; ...);
прямая (LDA addr; SHLD addr; IN port; ...).

Список команд. Команды МП КР580ИК80А приведены в табл. Г.3 — Г.5. Следует остановиться на следующих особенностях.

1. Трехбайтовые поля адресации источника и приемника информации кодируются в машинных командах символами SSS и DDD соответственно. В мнемонических изображениях двухадресных команд приемник указывается на первом месте, а источник — на втором.

2. В описаниях команд для обозначения содержимого регистра или ячейки памяти используется запись вида: (r1), (r), (H) (M) и т. п.

Таблица Г.2

Коды условий, используемые в командах условных переходов

Код (CCC)	Мнемоника (cc)	Условие	Код (CCC)	Мнемоника (cc)	Условие
000	NZ	Не нуль (Z=0)	001	Z	Нуль (Z=1)
010	NC	Нет переноса (C=0)	011	C	Перенос (C=1)
100	PO	Нечетность (P=0)	101	PE	Четность (P=1)
110	P	Плюс (S=0)	111	M	Минус (S=1)

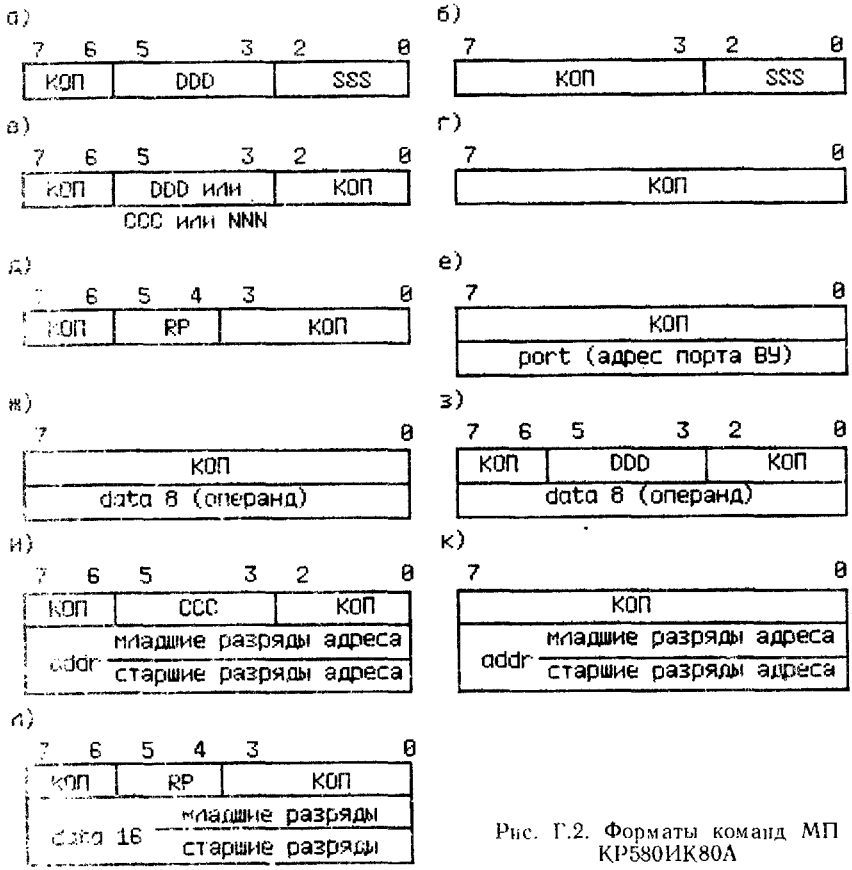


Рис. Г.2. Форматы команд МП КР580ИК80А

3. Формат каждой команды указан в табл. Г.3 – Г.5 в виде ссылки на соответствующий формат рис. Г.2.

Таблица Г.3

Список команд пересылки МП КР580ИК80А

Мнемоника	Двоичный код	Формат	Число тактов	Название и описание
				Пересылка
MOV r1, r2	01DDDSSS	а	5	(r2) ⇒ (r1)
MOV r, M	01DDD110	а	7	(M) ⇒ (r)
MOV M, r	01110SSS	а	7	(r) ⇒ (M)
MVI r, data 8	00DDD110	з	7	data ⇒ (r)
MVI M, data 8	00110110	з	10	data ⇒ (M)
				Загрузка
LXI rp, data 16	00RP0001	л	10	data ⇒ (rp)
LDA addr	00111010	к	13	(addr) ⇒ (A)
LHLD addr	00101010	к	16	(addr) ⇒ (L), (addr + 1) ⇒ (H)
LDAX rp	00RP1010	д	7	(rp) ⇒ (A)
				Обмен
XCHG	11101011	г	4	(H) ⇒ (D), (L) ⇒ (E)
				Запись
STA addr	00110010	к	13	(A) ⇒ (addr)
SHLD addr	00100010	к	16	(L) ⇒ (addr), (H) ⇒ (addr + 1)
STAX rp	00RP0010	д	7	(A) ⇒ (rp)

4. Дополнительные байты команд с форматами от «е» до «л» используются для хранения непосредственного операнда (data 8 или data 16), адреса ячейки памяти (addr) или порта внешнего устройства (port).

Таблица Г.4

Список арифметических и логических команд МП КР580ИК80А

Мнемоника	Двоичный код	Формат	Число тактов	Название и описание	Устанавливаемые признаки
				Сложение	
ADD r	10000SSS	б	4	(A) + (r) => (A)	
ADD M	10000110	б	7	(A) + (M) => (A)	S, Z, P,
ADI data 8	11000110	ж	7	(A) + data 8 => (A)	C, AC
				Сложение с переносом	
ADC r	10001SSS	б	4	(A) + (r) + (C) => (A)	
ADC M	10001110	б	7	(A) + (M) + (C) => (A)	S, Z, P,
ACI data 8	11001110	ж	7	(A) + data 8 + (C) => (A)	C, AC
				Вычитание	
SUB r	10010SSS	б	4	(A) − (r) => (A)	
SUB M	10010110	б	7	(A) − (M) => (A)	S, Z, P,
SUI data 8	11010110	ж	7	(A) − data 8 => (A)	C, AC
				Вычитание с заемом	
SBB r	10011SSS	б	4	(A) − (r) − (C) => (A)	
SBB M	10011110	б	7	(A) − (M) − (C) => (A)	S, Z, P,
SBI data	11011110	ж	7	(A) − data 8 − (C) => (A)	C, AC
				Логическое умножение	
ANA r	10100SSS	б	4	(A) AND (r) => (A)	
ANA M	10100110	б	7	(A) AND (M) => (A)	S, Z, P,
ANI data 8	11100110	ж	7	(A) AND data 8 => (A)	C, AC
				Исключающее ИЛИ	
XRA r	10101SSS	б	4	(A) XOR (r) => (A)	
XRA M	10101110	б	7	(A) XOR (M) => (A)	S, Z, P
XRI data	11101110	ж	7	(A) XOR data 8 => (A)	C, AC
				ИЛИ	
ORA r	10110SSS	б	4	(A) OR (r) => (A)	S, Z, P,
ORA M	10110110	б	7	(A) OR (M) => (A)	C = 0,
ORI data 8	11110110	ж	7	(A) OR data 8 => (A)	AC = 0
				Сравнение	
CMP r	10111SSS	б	4	(A) − (r)	
CMP M	10111110	б	7	(A) − (M)	S, Z, P,
CPI data 8	11111110	б	7	(A) − data 8	C, AC
				Увеличение	
INR r	00DDD100	в	5	(r) + 1 => (r)	S, Z, P
INR M	00110100	в	10	(M) + 1 => (M)	AC
INX rp	00RP0011	д	5	(rp) + 1 => (rp)	
				Уменьшение	
DCR r	00DDD101	в	5	(r) − 1 => (r)	S, Z, P
DCR M	00110101	в	10	(M) − 1 => (M)	AC
DCX rp	00RP1011	д	5	(rp) − 1 => (rp)	
				Сложение содержимых регистровых пар	
DAD rp	00RP1001	д	10	(H, L) + (rp) => (H, L)	C
RLC	00000111	г	4	Циклический сдвиг влево. Все биты A смещаются на один разряд влево. Старший разряд A переходит в его нулевой разряд и регистр признака C	C
RRC	00001111	г	4	Циклический сдвиг вправо. Выполняется аналогично RLC. Старший разряд A и C приобретают значение младшего разряда A	C

Мнемоника	Двоичный код	Формат	Число тактов	Название и описание	Устанавливаемые признаки
RAL	00010111	г	4	Циклический сдвиг влево через перенос Все биты A сдвигаются на один разряд влево; старший разряд A переходит в C, а C — в младший разряд A	C
RAR	00011111	г	4	Циклический сдвиг вправо через перенос Выполняется аналогично RAL, но сдвиг вправо	C
DAA	00100111	г	4	Десятичная коррекция аккумулятора Преобразование содержимого A в двоично-десятичный код	S, Z, P, C, AC
CMA	00101111	г	4	Инвертирование A Получение обратного кода (инверсии) содержимого A	--
STC	00110111	г	4	Установка C 1→C	C=1
CMC	00111111	г	4	Инвертирование C Инвертирование содержимого регистра C	C

Таблица Г.5

Список команд управления, ввода-вывода и работы со стеком

Мнемоника	Двоичный код	Формат	Число тактов	Название и описание
JMP addr	11000011	к	10	Безусловный переход к команде, адрес которой определяется вторым и третьим байтами данной команды
PCHL	11101001	г	5	Косвенный переход по адресу, указанному в регистрах H, L
Jcc addr	11CCC010	и	10	Условные переходы: если условие (cc из табл. Г.2) истинно, то переход к команде, адрес которой определяется вторым и третьим байтами данной команды; иначе выполняется команда, расположенная вслед за данной
CALL addr	11001101	к	17	Обращение к подпрограмме: содержимое счетчика команд заносится в стек по адресу, на который указывает SP; содержимое SP уменьшается на 2 и выполняется переход к команде, адрес которой определяется вторым и третьим байтами данной команды
Ccc addr	11CCC100	и	11/17	Условное обращение к подпрограмме: если условие (cc из табл. Г.2) истинно, то выполняется команда CALL; иначе выполняется команда, расположенная вслед за данной
RET	11001001	г	10	Возврат из подпрограммы: переход к команде, адрес которой записан в верхней паре ячеек стека, и увеличение содержимого SP на 2
Rcc	11CCC000	в	5/11	Условный возврат из подпрограммы: если условие (cc из табл. Г.2) истинно, то выполняется команда RET; иначе выполняется команда, расположенная вслед за данной

Продолжение табл. Г.5

Мнемоника	Двоичный код	Формат	Число тактов	Название и описание
RST п	11NNN111	в	11	Повторный запуск МП с адреса 8·NNN (0, 8, 16,...,56)
RUSH гр	11RP0101	д	11	Загрузка стека содержимым пары регистров (гр)
POP гр	11RP0001	д	10	Выдача данных из стека в пару регистров (гр); при выдаче PSW происходит установка S, Z, P, C и AC
XTHL	11100011	г	18	Обмен содержимым верхней пары ячеек стека и пары регистров H, L
SPHL	11111001	г	5	Пересылка содержимого пары регистров H, L в указатель стека SP
IN port	11011011	е	10	Ввод в аккумулятор данных из адресуемого порта
OUT port	11010011	е	10	Вывод данных из аккумулятора в адресуемый порт
EI	11111011	г	4	Разрешение прерывания
DI	11110011	г	4	Запрещение прерывания
HLT	01110110	г	7	Останов
NOP	00000000	г	4	Нет операции

5. Признаки результата (S, Z, P, C и AC) устанавливаются лишь при выполнении большинства арифметических и логических команд, а также команды POP PSW. В табл. Г. 4 для каждой из команд дан перечень устанавливаемых признаков (в командах INX, DCX и CMA признаки не устанавливаются).

6. В командах условного перехода, условного обращения к подпрограмме и условного возврата из подпрограммы используются коды условий (ССС) из табл. Г.2. Мнемонические обозначения этих команд составляются из символов J, C или R и соответствующих символов (сс) из табл. Г. 2 (например, JNC, CP, RM).

7. Число тактов, необходимых для исполнения команд условного обращения к подпрограмме и условного возврата из подпрограммы, зависит от того, выполнено (знаменатель дроби) или не выполнено (числитель дроби) условие, указанное в команде.

СПИСОК ЛИТЕРАТУРЫ

1. **Алексеенко А. Г., Галицын А. А., Иванников А. Д.** Проектирование радиоэлектронной аппаратуры на микропроцессорах: Программирование, типовые решения, методы отладки. М.: Радио и связь, 1984. 272 с.

2. **Алексеенко А. Г., Коломбет Е. А., Стародуб Г. И.** Применение прецизионных аналоговых микросхем. М.: Радио и связь, 1985. 304 с.

3. **Балашов Е. П., Григорьев В. Л., Петров Г. А.** Микро- и мини- ЭВМ: Учеб. пособие для вузов. Л.: Энергоатомиздат, 1984. 376 с.

4. **Бытовая** персональная микроЭВМ «Электроника БК-0010»/С. М. Косенков, А. Н. Полосин, З. А. Счепицкий и др.//Микропроцессорные средства и системы. 1985. № 1. С. 22—25.

5. **Васильев Н. П., Горовой В. Р.** Микропроцессоры: Аппаратурно-программные средства отладки: Учеб. пособие для втузов/Под ред. Л. Н. Преснухина. М.: Высшая школа, 1984. 95 с.

6. **Воробьев Н. В., Вернер В. Д.** Микропроцессоры: Элементная база и схемотехника средств сопряжения: Учеб. пособие для втузов/Под ред. Л. Н. Преснухина. М.: Высшая школа, 1984. 104 с.

7. **Вычислительная** техника социалистических стран: Сб. статей/Межправительств. комис. по сотрудничеству соц. стран в обл. вычисл. техники; Под общ. ред. Н. Е. Раковского. М.: Финансы и статистика, 1984. Вып. 16. 191 с.

8. **Гибсон Г., Лю Ю-Ч.** Аппаратные и программные средства микроЭВМ/Пер. с англ. В. Л. Григорьева; Под ред. В. В. Стаценка. М.: Финансы и статистика, 1983. 255 с.

9. **Гивоне Д., Россер Р.** Микропроцессоры и микрокомпьютеры: Вводный курс/Пер. с англ. М: Мир, 1983. 464 с.

10. **Гильмор Ч.** Введение в микропроцессорную технику/Пер. с англ. М.: Мир. 1984. 334 с.

11. **Горбунов В. Л., Панфилов Д. И.** Микропроцессоры: Лабораторный практикум: Учеб. пособие для втузов/Под ред. Л. Н. Преснухина. М.: Высшая школа, 1984. 104 с.

12. **Горбунов В. Л., Панфилов Д. И., Преснухин Д. Л.** Микропроцессоры: Основы построения микроЭВМ: Учеб. пособие для втузов/Под ред. Л. Н. Преснухина. М.: Высшая школа, 1984. 144 с.

13. **Заморин А. П., Мячев А. А., Селиванов Ю. П.** Вычислительные машины и комплексы: Справочник/Под ред. Б. Н. Наумова, В. В. Пржиялковского. М.: Энергоатомиздат, 1985. 264 с.

14. **Каган Б. М.** Электронные вычислительные машины и системы: Учеб. пособие для вузов. М.: Энергоатомиздат, 1985. 552 с.

15. **Кобылинский А. В., Ликовецкий Г. П.** Однокристальные микроЭВМ серии К1816// Микропроцессорные средства и системы. 1986. № 1. С. 10—19.

16. **Кобылинский А. В., Москалевский А. И., Темченко В. Л.** Однокристальный высокопроизводительный 16-разрядный микропроцессор КМ1810ВМ86//Микропроцессорные средства и системы. 1986. № 1. С. 28—33.

17. **Королев Л. Н.** Структуры ЭВМ и их математическое обеспечение. М.: Наука, 1978. 345 с.

18. **Коффрон Дж.** Технические средства микропроцессорных систем: Практический курс/Пер. с англ. М.: Мир, 1983. 344 с.

19. **Левенталь Л.** Введение в микропроцессоры: Программное обеспечение, аппаратные средства, программирование/Пер. с англ. М.: Энергоатомиздат, 1983. 464 с.

20. **Магистрально-модульные** микропроцессорные системы/Б. М. Малашевич, В. Л. Дшхунян, Ю. Н. Борщенко и др.//Микропроцессорные средства и системы. 1984. № 4. С. 3—11.

21. **Майоров С. А., Новиков Г. И.** Структура электронных вычислительных машин. Л.: Машиностроение, 1979. 384 с.

22. **Малые** ЭВМ и их применение/ Ю. А. Дедов, М. А. Островский, К. В. Песелев и др.; Под общ. ред. Б. Н. Наумова. М.: Статистика, 1980. 231 с.

23. **Методические** указания к лабораторным работам по базовой ЭВМ. Л.: Ин-т точной механики и оптики, 1986. 54 с.

24. **Микропроцессорные** комплекты интегральных схем: Состав и структура: Справочник/В. С. Борисов, А. А. Васенкова, Б. М. Малашевич и др.; Под ред. А. А. Васенкова, В. А. Шахнова. М.: Радио и связь, 1982. 192 с.

25. **МикроЭВМ**/Пер. с англ. под ред. А. Дирксена. М.: Энергоатомиздат, 1982. 328 с.

26. **МикроЭВМ** СМ-1800: Архитектура, программирование, применение/А. В. Гиглавый, Н. Д. Кабанов, Н. Л. Прохоров, А. Н. Шкамарда. М.: Финансы и статистика, 1984. 134 с.

27. **МикроЭВМ** «Электроника С5» и их применение/М. П. Гальперин, В. Я. Кузнецов, Ю. А. Масленников и др.; Под ред. В. М. Пролейко. М.: Сов. радио, 1980. 157 с.

28. **Нестеров П. В.** Микропроцессоры: Архитектура и ее оценка: Учеб. пособие для втузов/Под ред. Л. Н. Преснухина. М.: Высшая школа, 1984. 104 с.

29. **Однокристальные** 4-разрядные микроЭВМ серии КР1820/В. А. Бобков, Ю. Н. Бурмистров, В. А. Кособрюхов и др.//Микропроцессорные средства и системы. 1986. № 1. С. 19—27.

30. **Однокристальные** 4-разрядные микроЭВМ серии К1814/В. Н. Златопольский, И. Е. Лобов, А. И. Стоянов, И. А. Шадрин//Микропроцессорные средства и системы. 1985. № 1. С. 3—10.

31. **Одноплатные** микроконтроллеры «Электроника С5-41»/М. П. Гальперин, А. В. Гинтер, В. В. Городецкий и др.//Микропроцессорные средства и системы. 1984. № 2. С. 20—23.

32. **Одноплатные** микроЭВМ ряда «Электроника МС1201»/В. Л. Дшхунян, Ю. Н. Борщенко, Ю. Л. Отрохов, С. А. Шишарин//Микропроцессорные средства и системы. 1985. № 2. С. 8—13.

33. **Проектирование** цифровых систем на комплектах микропрограммируемых БИС/С. С. Булгаков, В. М. Мещеряков, В. В. Новоселов, А. А. Шумилов; Под. ред. В. Г. Колесникова. М.: Радио и связь, 1984. 240 с.

34. **Профессиональные** персональные ЭВМ «Искра 226»/С. Н. Абрамович, В. В. Бойко, Б. П. Бутрин и др.//Микропроцессорные средства и системы. 1985. № 2. С. 29—36.

35. **Прохоров Н. Л., Ландау И. Я.** МикроЭВМ СМ-1800 и ее программное обеспечение//Микропроцессорные средства и системы. 1984. № 2. С. 28—30.

36. **Симонс Дж.** ЭВМ пятого поколения: компьютеры 90-х годов/Пер. с англ.: Предисл. В. А. Мясникова. М.: Финансы и статистика, 1985. 173 с.

37. **Соучек Б.** Микропроцессоры и микроЭВМ/Пер. с англ. под ред. А. И. Петренко. М.: Сов. радио, 1979. 520 с.

38. **Таненбаум Э.** Многоуровневая организация ЭВМ/Пер. с англ. под ред. В. К. Потопкого и М. Б. Игнатьева. М.: Мир, 1979. 547 с.

39. **Трейстер Р.** Персональный компьютер фирмы ИБМ/Пер. с англ. М.: Мир, 1986. 208 с.

40. **Труды** института инженеров по электронике и радиоэлектронике (ТИИЭР)/Пер. с англ. М.: Мир, 1984. Т. 72. № 3 (Тематический вып. «Персональные ЭВМ»). 187 с.

41. **Уокерли Дж.** Архитектура и программирование микроЭВМ: В 2-х кн./Пер. с англ. М.: Мир, 1984. Кн. 1. 486 с.; Кн. 2. 341 с.

42. **Фролов Г. И., Гембицкий Р. А.** Микропроцессоры: Автоматизированные системы контроля объектов/Под ред. Л. Н. Преснухина. М.: Высшая школа, 1984. 87 с.

43. **Хвощ С. Т., Варлинский Н. Н., Попов Е. А.** Микропроцессоры и микроЭВМ в системах автоматического управления: Справочник. Л.: Машиностроение, 1987. 640 с.

44. **Шаньгин В. Ф., Костин А. Е.** Микропроцессоры: Организация вычислительных процессов на микроЭВМ/Под ред. Л. Н. Преснухина. М.: Высшая школа, 1984. 119 с.

45. **Шилейко А. В., Шилейко Т. Ч.** Микропроцессоры. М.: Радио и связь, 1986. 112 с.

46. **Якубайтис Э. А.** Информационно-вычислительные сети. М.: Финансы и статистика, 1984. 232 с.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

А адаптер 107
адрес 19
адресация:
абсолютная 291
автодекрементная 289
— косвенная 289
автоинкрементная 289
— косвенная 289
косвенная 40, 49
косвенно-относительная 292
непосредственная 289
неявная 27
относительная 291
по базе 289
— — с индексированием 291
прямая 39
регистровая 289
— косвенная 289
с индексированием (индексация) 289
— — косвенная (косвенная индексация) 289
адресная шина 105
аккумулятор 21, 27, 33
алгоритм 5
— LIFO 109
алфавитно-цифровое печатающее устройство (АЦПУ) 145
аналоговая вычислительная машина (АВМ) 6
аналого-цифровой преобразователь (АЦП) 150, 152
аппаратные средства ЭВМ 100
арифметико-логическое устройство (АЛУ) 10, 27, 33, 80
архитектура ЭВМ 100
асинхронный обмен 68, 70
асинхронная последовательная передача данных 138
ассемблер 77, 265

Б базовая ЭВМ 31
байт 8, 15
банк памяти 126
бит 8, 15
— контроля четности (нечетности) 139
— стартовый 139
— стоповый 139
буферные усилители 128

В вектор прерываний 187
векторная приоритетная система прерываний 209
вентильные схемы 10
вершина стека 109
винчестерские диски 160
виртуальная ЭВМ 77
внешние устройства (ВУ) 61, 136
восьмеричные числа 15, 279
время восстановления 122
— доступа 122
— записи 122
выбор кристалла (CE) 126
выборка адреса 57
— команды 28, 56
вычитание 45

Г генератор синхроимпульсов 139
генератор тактовых импульсов (ГТИ) 10, 56
горизонтальная микрокоманда 88

Д двоичные числа, 7, 40, 279
двоично-десятичные числа 16
джойстик 273
дешифратор 97
— адреса 69, 125
— приказов 69
— столбцов 125
— строк 125
динамическая память 129
дискета 156
дисковая операционная система (ДОС) 271
дисковое запоминающее устройство 156
дисплей 148
дистанционная связь 141
дополнение 41
дополнительный код 41
драйвер (буферный усилитель) 128
— ввода-вывода 268
дуплексная линия связи 142

Е емкость памяти 122

З загрузчик 268
запоминающее устройство (ЗУ) 122
— — на магнитных дисках 156
— — магнитной ленте 160
— — оптическом диске 162
— — — цилиндрических магнитных доменах (ЦМД) 161
— оперативное (ОЗУ) 123
— — постоянное (ПЗУ) 123, 131
— — — программируемое (ППЗУ) 123, 131
— — — со стиранием (СППЗУ) 123, 131
— — — репрограммируемое 131
— — с последовательной выборкой 123
— — — произвольной выборкой (ЗУПВ) 123

— электрически изменяемое (ЭИПЗУ) 132
— — энергозависимое 124
— — энергонезависимое 124
запоминающий элемент (ЗЭ) 19, 124
— — динамический 129
— — статический 129
запрос на прерывание 72, 184

И ИЛИ (OR) 283, 288
исключающее (XOR) 284, 287, 288
инвертирование 42
инвертор 80
индексные ячейки 31, 50
инструментальная ЭВМ 269
интегральные схемы (ИС) 101, 118
— — большой степени интеграции (БИС) 102
— — сверхбольшой степени интеграции (СБИС) 103, 121
интерпретатор базовой ЭВМ 90, 92
интерпретация 76
интерфейс 107, 163
— microbus 166
— Q-bus 168
— радиальный параллельный (ИРПР) 173
— — последовательный (ИРПС) 173
— системный 107
исполнение команды 28, 59

К клавишный регистр (КР) 61
код:
обмена информацией (КОИ) 137, 280
операции (КОП) 20
Хемминга 133
кодирование:
информации 7, 13, 136
команд 14, 36
символов 14, 137, 280
чисел 13
команды 14, 20
— адресные 39
— безадресные 27, 39
— ввода-вывода 26, 39
— одноадресные 23, 27
— пересылки 26
— перехода 25
комбинационная схема 82
компаратор 128
компиляция 76
контроллер 66, 107, 170
— ПДП 189, 211
— прерываний 68, 71, 209
— программируемый 171
контроль четности (нечетности) 133, 139
кристалл 101, 118
кросс-ассемблер 267, 270

Л легирование 119
листинг 267
логическое умножение 50
локальные вычислительные сети 256

М магистральные приемопередатчики (формирователи) 128
макроассемблер 266
макрокоманда 266
машинный цикл 56
метабит 14
микрокалькулятор 10, 103
микрокоманда 28, 78
вертикальная 97
— горизонтальная 88
— операционная 78, 88
— управляющая 78, 88
микроконтроллер 103, 246
микрооперация 77
микропрограмма 28, 91
микропрограммирование 96
микропрограммное устройство управления (МПУ) 31, 78
микропроцессор (МП) 102
микропроцессорный комплект интегральных схем (МПК) 203
микроЭВМ 103, 245
— автономные 246, 250
— встраиваемые 246
— персональные 246, 253
мини-ЭВМ 102
мнемоническое кодирование 36
многоуровневые ЭВМ 77
моделирующая программа 270
модем 142
модуль (банк) памяти 126
модуляция 141
монитор 271
МОП-технология 121
мультиплексированная шина 105
мультиплексор 125
«Мышь» 273

Н накопитель на гибком магнитном диске (НГМД) 157
— — магнитной ленте (НМЛ) 160
— — — — кассетный (КНМЛ) 160
— магнитном диске (НМД) 156
нанопамять 98

О обмен:
асинхронный 68, 70
по прерыванию 68, 71
синхронный 67
операнд 20
оперативное запоминающее устройство (ОЗУ) 123
операционная система (ОС) 101, 271
операция ввода 67
— вывода 67
ошибка кадра 140

П память 19, 31
— адресная 19, 31, 108
— ассоциативная 110
— динамическая 129
— микрокоманд 78, 89
— статическая 129
— стековая 109
передача данных:
дуплексная 142
между регистрами 79
параллельная 138, 174
полудуплексная 142
последовательная 138, 178
— асинхронная 138, 180
— синхронная 138, 178

при прямом доступе к памяти 67, 189
программно-управляемая 67
симплексная 142
ПДП с блокировкой процессора 192
— захватом цикла 190
периферийные устройства 136
персональный компьютер 103, 253, 272
перфолентоные ВУ 143
печатающие устройства 144
печатные платы 116
пиксел 148
планарная технология 118, 121
подпрограмма 53
— обработка прерываний 71, 185
порт ввода вывода 171
прерывание программы 68, 71, 184
признак результата 83
приемопередатчики (формирователи):
магистральные 128
шинные 128
прототип 269
процессор 18, 27, 31
— с аккумулятором 111
— с регистрами общего назначения 112
со стековой организацией 113
прямой доступ к памяти (ПДП) 189

Р рабочий такт 13, 56, 78
разрешение прерывания 71, 185
регенерация динамической памяти 130
регистр 8
— адреса (РА) 23, 27, 32
— буферный (БР) 80
данных (РД) 25, 27, 33
клавишный (КР) 61
— команд (РК) 27, 32
— микрокоманд (РМК) 78, 89
общего назначения (РОН) 112
переноса (С) 33
— разрешения прерывания 185
— состояний (РС) 27, 86, 185
редактор:
связей 268
текста 267, 274

С сдвиг:
арифметический 285
циклический 53, 285, 295, 296
символическое кодирование 36
синхронизация 10
слово 8
— состояния программы (ССП или PSW) 292
стек 109
счетчик:
команд (СК) 25, 27, 32
микрокоманд (СчМК) 78, 89
сумматор 80

Т тактовые импульсы 10
такт рабочий 13, 56, 78
транслятор 5
требование прерывания 184
триггер 129
тристабильные усилители мощности 128

У указатель:
поля 23
стека 109
устройства:
ввода 19, 66
вывода 19, 66
управления 19, 28, 78

Ф фазы (подтакты) 98
файл 271, 272
флаг (флажок) 67
форматирование диска 158
форматы команд 37, 282, 294,
формирователи шинные 128
фотолитография 121
фоторезист 117

Ц цикл:
записи 123
команды 56
машинный 56
памяти 123
считывания 123
циклический сдвиг 53, 285, 291, 292
цилиндрические магнитные доменты (ЦМД) 161
цифроаналоговый преобразователь (ЦАП) 151

Ч число:
восьмеричное 15, 279
двоично-десятичное 16
двоичное 7, 40, 279
с плавающей запятой 13
целое без знака 13, 40
со знаком 41
шестнадцатеричное 15, 279, 280

Ш шины 10
— двунаправленные 105
— изолированные 105
мультиплексируемые 105
общие 105
раздельные 105

Э электронная вычислительная машина (ЭВМ) 7
элемент памяти 8, 19, 124
изображения (ЭИ) 148
эмулятор 162, 269, 270
эмуляторы внешних ЗУ (электронный диск) 162
эпитаксиальный слой 119

Я язык ассемблера 77, 265
языки высокого уровня 265, 269
ячейка памяти 9, 19

ОГЛАВЛЕНИЕ

Предисловие 3

Часть I

Основы функционирования и построения ЭВМ

Глава 1

Общие сведения о представлении и обработке информации в ЭВМ 5

1.1. Две формы представления информации . . . . . —
1.2. Способы представления дискретной информации 7
1.3. Системы счисления, используемые в вычислительной технике 15
1.4. Структура и принцип функционирования ЭВМ 18

Глава 2

Базовая ЭВМ 31

2.1. Назначение и структура базовой ЭВМ . . . . . —
2.2. Кодирование программ и система команд 36
2.3 Арифметические операции . . . 40
2.4. Управление вычислительным процессом, сдвиги и логические операции 46
2.5 Подпрограммы и параметры . . . 53
2.6. Выполнение машинных команд . . 56

Глава 3

Организация ввода-вывода информации и базовой ЭВМ 61

3.1. Отладочный пульт базовой ЭВМ —
3.2. Устройства ввода-вывода базовой ЭВМ 66
3.3. Программно-управляемая передача информации 67
3.4. Асинхронный обмен 70
3.5. Обмен по прерыванию программы 71

Глава 4

Микропрограммное устройство управления базовой ЭВМ 76

4.1. Многоуровневые ЭВМ и их микропрограммный уровень 79
4.2. Компоненты процессора и основные операции 87
4.3. Микропрограммное управление вентильными схемами 90
4.4. Интерпретатор базовой ЭВМ 96
4.5. Другие варианты построения микрокоманд 100

Глава 5

Архитектура микроЭВМ 104

5.1. Общие положения 108
5.2. Структуры типичных микроЭВМ . 111
5.3. Организация памяти 116
5.4. Структуры процессоров . . . . —
5.5. Немного о конструкции микроЭВМ 122

Глава 6

Подсистема памяти 124

6.1. Характеристики запоминающих устройств 129
6.2. Конфигурация ЗУ с произвольным доступом 132
6.3. Оперативные и постоянные ЗУ с произвольным доступом . . . 136
6.4. Надежность запоминающих устройств . . . . . —

Глава 7

Периферийные устройства микроЭВМ 142

7.1. Общие сведения о периферийных устройствах и их связи с микроЭВМ 150
7.2. Устройства связи человек — микроЭВМ 154
7.3. Устройства для связи микроЭВМ с объектами управления . . 163
7.4. Внешние запоминающие устройства большой емкости . . . . . —

Глава 8

Организация ввода-вывода информации 170

8.1. Интерфейсы микроЭВМ . . . 174
8.2. Контроллеры внешних устройств 178
8.3. Параллельная передача данных 184
8.4. Последовательная передача данных 189
8.5 Организация прерываний в микроЭВМ 303
8.6. Организация прямого доступа к памяти 303

Глава 9

Моделирование процессов функционирования микроЭВМ 192

9.1. Модельный подход при изучении устройств микропроцессорной техники . . . . . .

9.2. Взаимодействие с функциональной моделью 194

9.3. Построение функциональной модели 199

9.4. Исследования, проводимые на функциональной модели 200

Часть II

Современные средства микропроцессорной техники и перспективы их развития

Глава 10

Микропроцессорные комплекты интегральных схем 203

10.1. Классификация микропроцессорных комплектов интегральных схем . . . . . —

10.2. Микропроцессорный комплект БИС КР580 204

10.3. Микропроцессорный комплект БИС К1810 215

10.4. Микропроцессорный комплект БИС К1801 218

10.5. Микропроцессорный комплект БИС К1804 227

10.6. Пример применения микропроцессорного комплекта БИС КР580 . . 238

Глава 11

Современные микроЭВМ 245

11.1. Классификация микроЭВМ . . . —

11.2. Встраиваемые микроЭВМ . . . 246

11.3. Автономные микроЭВМ 250

11.4. Персональные микроЭВМ . . . 253

11.5. Локальные вычислительные сети микроЭВМ 256

11.6. Примеры применения микро «Электроника-60» . . . .

Глава 12

Программное обеспечение микро

12.1. Общие сведения . . . .

12.2. Программное обеспечение м контроллеров и встраива микроЭВМ . . . . . .

12.3. Программное обеспечение номных микроЭВМ . . . .

12.4. Программное обеспечение н нальных ЭВМ . . . . .

Глава 13

Тенденции развития ЭВМ . . .

13.1. Поколения ЭВМ и эпохал события в развитии вычисли ной техники . . . . . .

13.2. Классификация ЭВМ и тенд их развития . . . . . .

13.3. На пути к вычислительным с мам пятого поколения . . .

Приложения . . . . . .

Приложение А

Таблицы перевода чисел из одно стемы счисления в другую . . .

Приложение Б

Таблицы кодирования информаци редаваемой между отдельными со ляющими вычислительных систем

Приложение В

Система команд микроЭВМ «Эле ника-60» . . . . . .

Приложение Г

Система команд микропроце КР580ИК80А . . . . . .

Список литературы . . . . .

Предметный указатель . . . .