И.В. Хмелевский, В.П. Битюцкий

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

ОДНОПРОЦЕССОРНЫЕ ЭВМ

ЧАСТЬ 2

Федеральное агентство по образованию
ГОУ ВПО «Уральский государственный технический университет-УПИ»

И.В. Хмелевский, В.П. Битюцкий

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

ОДНОПРОЦЕССОРНЫЕ ЭВМ

ЧАСТЬ 2

Конспект лекций

Издание второе, исправленное и дополненное

Научный редактор
проф., д-р техн.наук Л.Г. Доросинский

Екатеринбург 2005

УДК 681.3
ББК 32.973.202я73
Х-65

Рецензенты:
кафедра информатики УГГУ (зав. кафедрой доц. канд.техн.наук А.В.Дружинин);
доц., канд. техн. наук Г.Б.Захарова (УрО РАН)

Авторы: И.В.Хмелевский, В.П.Битюцкий

Х-65 **Организация ЭВМ и систем. Однопроцессорные ЭВМ. Часть 2.:** Конспект лекций / И.В. Хмелевский, В.П. Битюцкий. 2-е изд., испр. и допол. Екатеринбург: ГОУ ВПО УГТУ-УПИ, 2005. 98 с.

ISBN 5-321-00534-6

Настоящий конспект лекций продолжает материал, изложенный в первой части. Конспект посвящен изучению основ организации и функционирования ЭВМ в целом и ее отдельных узлов, взаимодействия ЭВМ и периферийных устройств, в том числе многопроцессорных систем, перспективных направлений в развитии вычислительной техники, приобретению опыта разработки простейших микропроцессорных устройств.

Конспект предназначен для студентов всех форм обучения направления 230100 – Информатика и вычислительная техника

Табл. 4, Рис. 61.

Подготовлено кафедрой "Автоматика и информационные технологии".

УДК 681.3
ББК 32.973.202я73

ISBN 5-321-00516-8

ОГЛАВЛЕНИЕ

4. ПРИНЦИПЫ ПОСТРОЕНИЯ УСТРОЙСТВ ВНУТРЕННЕЙ ПАМЯТИ

Памятью ЭВМ называют совокупность устройств, служащих для запоминания, хранения и выдачи информации. Отдельные устройства, входящие в эту совокупность, называются запоминающими устройствами или памятями того или иного типа. В настоящее время и ЗУ, и память стали практически синонимами.

Производительность ЭВМ и ее возможности в большой степени зависят от характеристик ЗУ, причем в любой ЭВМ общего назначения используют несколько типов ЗУ.

Основные операции:

- *Запись* – занесение информации в память;
- *Считывание* – выборка информации из памяти.

Обе этих операции называются в общем случае *обращением к памяти*.

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

Коротко рассмотрим важнейшие характеристики ЗУ – емкость, удельную емкость, быстродействие, которые характерны для любых типов ЗУ, а также некоторые методы их классификации.

- *Емкость памяти* определяется максимальным количеством данных, которые могут в ней храниться одновременно. Емкость измеряется в битах, байтах, машинных словах (об этом говорилось в самом начале курса). Используют обычно более крупные единицы измерения: 1К = 1024 (Кбит, Кбайт, Кслов), 1024 Кбайт = 1 Мбайт, 1024 Мбайт = 1 Гбайт.
- *Удельная емкость* определяется как отношение емкости ЗУ к его физическому объему и характеризует степень технологического совершенства ЗУ.
- *Удельная стоимость* определяется как отношение стоимости ЗУ к его емкости и определяет, помимо технологического совершенства конкурентоспособность изделия на рынке.
- *Быстродействие памяти* определяет продолжительность операции обращения и делится:
  - на время обращения при *считывании* $t_{обр}^{сч}$. Это время, необходимое на поиск нужной единицы информации и ее считывание;
  - время обращения при записи $t_{обр}^{зап}$. Это время, необходимое для поиска места для хранения данной единицы информации и ее запись в память.

В некоторых устройствах памяти считывание информации сопровождается ее разрушением (стиранием). В этом случае цикл обращения к ЗУ при считывании должен содержать операцию *регенерации* считываемой информации на прежнем месте в памяти. В ряде случаев ЗУ требуют перед началом записи привести запоминающие элементы в некоторое начальное состояние. В этом случае цикл обращения к ЗУ при записи должен содержать операции подготовки запоминающих элементов к самой операции записи.

В общем случае продолжительность цикла обращения к памяти при считывании состоит из следующих компонент:

$$t_{обр}^{сч} = t_{дост}^{счит} + t_{счит} + t_{рег},$$

где $t_{дост}^{счит}$ – время доступа при считывании – интервал времени между началом операции обращения при считывании и моментом, когда доступ к данной единице информации стал возможен;

$t_{счит}$ – продолжительность самого физического процесса считывания, т.е. процесса обнаружения и фиксации состояния соответствующих запоминающих элементов или участков поверхности носителя;

$t_{рег}$ – время на восстановление разрушенной при считывании информации. В ЗУ без разрушения $t_{рег} = 0$.

Продолжительность цикла обращения к памяти при записи информации в общем случае состоит из следующих компонент:

$$t_{обр}^{зап} = t_{дост}^{зап} + t_{подг} + t_{зап},$$

где $t_{дост}^{зап}$ – время доступа при записи – интервал времени от начала обращения до момента, когда становится возможен доступ к запоминающему элементу или участку поверхности носителя, в который производится запись;

$t_{подг}$ – интервал времени, необходимый для приведения в исходное состояние запоминающих элементов или участков поверхности носителя для записи данной единицы информации;

$t_{зап}$ – продолжительность самого физического процесса записи, т.е. время для изменения физического состояния запоминающих элементов или участка поверхности носителя.

В большинстве случаев

$$t_{дост}^{сч} = t_{дост}^{зап} = t_{дост}.$$

В качестве продолжительности *цикла обращения к памяти* принимается величина

$$t_{обр} = \max\left(t_{обр}^{сч}, t_{обр}^{зап}\right).$$

Следует иметь в виду, что для хранения информации в ЭВМ используются устройства памяти, построенные на разных принципах действия и имеющие разнообразнейшие технические и конструктивные реализации. Кроме того, все они являются сложнейшими электронными устройствами. Поэтому устройства памяти обладают многочисленными характеристиками, по каждой из которых можно производить классификацию устройств. Ниже будут рассмотрены только основные критерии, по которым принято квалифицировать ЗУ, а именно: по принципу действия, по реализации в памяти операций обращения, по способу доступа к хранимой информации.

1. Принцип действия. По этому признаку основные типы ЗУ, наиболее широко используемые в современных ЭВМ, делятся:

- на электронные, в которых в качестве запоминающих элементов используют полупроводники;
- магнитные с неподвижными запоминающими элементами;
- магнитомеханические с движущимися магнитными носителями информации;
- оптические с движущимся носителем информации;
- магнитооптические с движущимся носителем информации.

В качестве внутренней памяти ЭВМ в абсолютном большинстве случаев используются электронные ЗУ на полупроводниковых элементах. В редких случаях в специализированных управляющих ЭВМ используются ЗУ с неподвижными магнитными запоминающими элементами. Остальные типы устройств используются в качестве внешних памятей ЭВМ. Более подробно эти устройства рассматриваются в отдельном курсе "Периферийные устройства ЭВМ".

В настоящее время разрабатываются и исследуются многочисленные "нетрадиционные" ЗУ, а именно: ЗУ на приборах с зарядовой связью, акустоэлектронные ЗУ, пьезоэлектронные ЗУ, магнитоэлектронные ЗУ и т.д.

2. Способ реализации в памяти операций обращения. По этому признаку различают:

- Память с произвольным обращением, допускающую как считывание, так и запись информации (RAM). Это энергозависимые ЗУ (информация в них сохраняется только при наличии питания), которые используются для построения ОЗУ, кэш, СОП и т.д.
- Память постоянная, допускающая только считывание информации, заложенной в нее в процессе изготовления или настройки (ROM). Это энергонезависимые ЗУ (информация в них сохраняется при отсутствии питания), которые, в свою очередь, делятся на постоянные ЗУ (ПЗУ, EPROM) и перепрограммируемые ЗУ (ППЗУ, EEPROM). Быстродействие RAM и ROM примерно одинаковое.
- Флэш ППЗУ (Flash EEPROM) – энергонезависимые перепрограммируемые ЗУ, информация в которых сохраняется до нескольких лет. Обращения к ним возможно как для записи, так и для чтения. Однако быстродействие этих ЗУ ниже, чем у RAM и ROM. Обычно флэш используются для накопления информации. Число перезаписей флэш ограничено.

3. Способ организации доступа. По этому признаку различают ЗУ с непосредственным (произвольным), с прямым (циклическим) и последовательным доступом.

- Непосредственный (произвольный) доступ. В ЗУ этого типа время доступа, а поэтому и цикл обращения не зависят от места расположения элемента памяти, с которого производится считывание или в который записывается информация. В большинстве случаев это электронные ЗУ, в которых непосредственный доступ реализуется с помощью электронных логических схем. В ЗУ с произвольным доступом цикл обращения составляет от 1-2 мкс до единиц наносекунд.

Независимость $t_{обр}$ от положения запоминающего элемента в запоминающем массиве имеет место только до определенной частоты обращений процессора к ЗУ. При увеличении частоты обращений до единиц наносекунд начинает сказываться геометрическое положение запоминающего элемента в массиве. Это обусловлено, прежде всего, конечной скоростью распространения электрического сигнала в изолированном проводнике, которая составляет примерно 60 % от скорости света.

Число разрядов, считываемых или записываемых в память с произвольным доступом параллельно во времени за одну операцию обращения, называется *шириной выборки*.

В других типах ЗУ используют более медленные электромеханические процессы, поэтому и цикл обращения больше.

- Прямой (циклический) доступ. К ЗУ этого типа относятся устройства на магнитных, оптических и магнитооптических дисках, а также на магнитных барабанах (последние в настоящее время используются очень редко). Благодаря непрерывному вращению носителя информации возможность обращения к некоторому участку носителя для считывания или записи циклически повторяется. В таких ЗУ время доступа составляет обычно от долей секунды до единиц миллисекунды.
- Последовательный доступ. К ЗУ этого типа относятся устройства на магнитных лентах. В процессе доступа производится последовательный просмотр участков носителя информации, пока нужный участок не займет некоторое исходное положение. Время доступа в худшем случае составляет минуты, поскольку магнитофон будет вынужден осуществить перемотку всей кассеты.

4.1. СТРУКТУРА ПАМЯТИ ЭВМ

Классическая пятиблочная структура Неймана, рассмотренная ранее, предполагала наличие только одного устройства памяти – ОП. Однако современные ЭВМ имеют иерархическую структуру памяти, каждый уровень которой характерен различным быстродействием и емкостью. Появление многочисленных иерархически расположенных уровней памяти связано, прежде всего, с постоянным увеличением разрыва в быстродействии процессора и ОП, которое необходимо скомпенсировать для повышения производительности ЭВМ в целом.

Кроме того, развитие программного обеспечения и расширение круга задач, решаемых на ЭВМ, требовали постоянного увеличения объема ОП. Между тем известно, что на всем протяжении развития ЭВМ требования к емкости и быстродействию ЗУ были противоречивы – чем выше быстродействие, тем технически труднее и дороже обходится увеличение емкости. Необходимость поддержания стоимости памяти ЭВМ на приемлемом уровне, а также множество технических проблем, связанных с построением быстродействующих ЗУ большого объема, и привели в процессе эволюции к созданию иерархической структуры памяти современной ЭВМ.

Несмотря на существенные различия в принципах функционирования и технической реализации различных уровней памяти, существуют общие принципы построения всей иерархии:

- чем ближе уровень памяти к процессору, тем выше его быстродействие и меньше емкость;
- алгоритмы взаимодействия всех уровней памяти построены так, что количество обращений верхнего, более быстродействующего уровня к нижележащему, менее быстродействующему, соседнему уровню является минимальным;
- обмен информацией между соседними иерархическими уровнями памяти в большинстве случаев осуществляется блоками фиксированной длины, что позволяет ускорить обмен за счет аппаратной реализации алгоритмов.

В общем случае память современной ЭВМ включает в себя следующие иерархические уровни:

- Сверхоперативная память (СОП), которая называется еще местной памятью.
- Кэш-память, которая обычно отсутствует в простейших процессорных устройствах. В более сложных ЭВМ кэш имеет несколько уровней, причем кэш верхнего уровня всегда находится в кристалле процессора.
- Оперативная (основная) память (ОП) или оперативное запоминающее устройство (ОЗУ), а также системное ПЗУ, объединенное с ОЗУ общим полем адресов.
- Память с прямым доступом на магнитных дисках.
- Память с последовательным доступом на магнитных лентах.

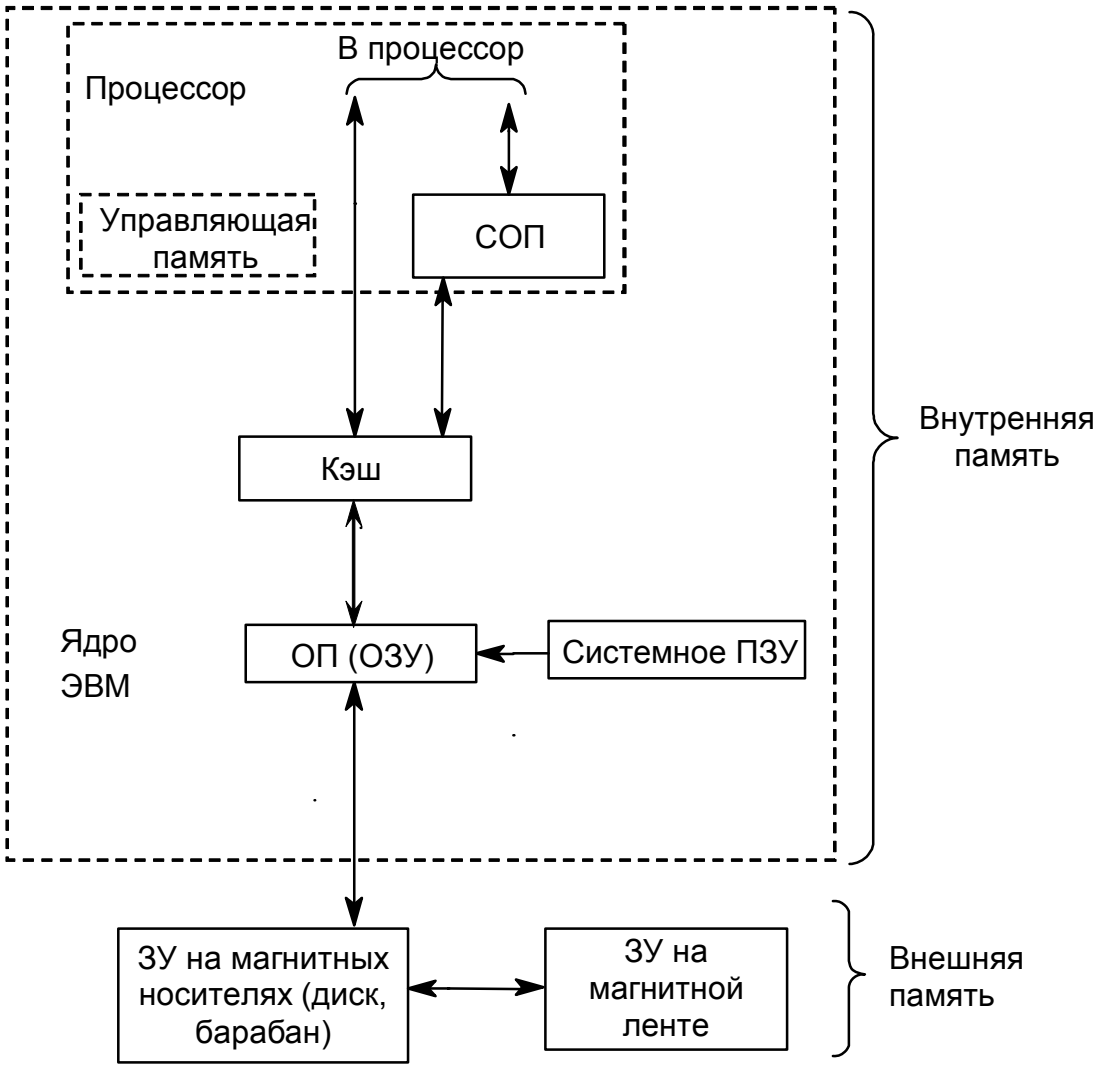
Устройства перечислены в порядке убывания быстродействия и увеличения объема.

Рассмотрим в самых общих чертах функциональное назначение устройств памяти, изображенных на рис. 4.1.

Оперативная (основная) память, системное ПЗУ. Название этого устройства памяти (ОП) отражает тот факт, что процессор может работать только с программами, которые загружены в ОП. Этот принцип был положен в основу функционирования первых однозадачных ЭВМ. По этому же принципу функционируют современные многозадачные однопроцессорные системы (многопроцессорные системы рассмотрены в последней части настоящего курса). При отсутствии кэш ОП служит для хранения информации, непосредственно используемой в вычислительном процессе. Из ОП в процессор поступают операнды и команды, а обратно – результаты выполненных операций.

Характеристики ОП непосредственно влияют на характеристики ЭВМ в целом и прежде всего на производительность (даже при наличии кэш).

Объем ОП зависит от целевого назначения ЭВМ и колеблется в очень широком диапазоне – от десятков Кбайт в простейших контроллерах до сотен Мбайт. В современных ЭВМ ОП всегда выполняется на полупроводниковых ЗУ и имеет длительность цикла обращения не более 1-2 мкс. (В ЭВМ первого поколения ОП строилась сначала на электронных лампах, а затем на ферритовых кольцах).



ЗУ- запоминающее устройство;
СОП – сверх оперативная память;
ОП (ОЗУ) – оперативная (основная) память;
оперативно запоминающее устройство;
ПЗУ – постоянно запоминающее устройство

Рис. 4.1. Иерархическая структура памяти

Системное ПЗУ имеет с ОП (ОЗУ) общее адресное пространство. Его объем и заполнение существенно зависят от целевого назначения ЭВМ.

Системное ПЗУ может хранить ядро операционной системы, утилиты, драйверы, служебные и прикладные программы и т.д. При включении ЭВМ или ее работе программы, записанные в системном ПЗУ, в большинстве случаев загружаются в ОП (ОЗУ) и только после этого обрабатываются процессором.

Сверхоперативная память. Необходимость в СОП возникла уже в первых ЭВМ, когда скорость работы процессора превысила скорость работы ОП. Современные СОП всегда строятся на полупроводниках и представляют собой наборы регистров, находящихся внутри кристалла процессора в непосредственной близости от АЛУ и УУ. Быстродействие СОП должно соответствовать быстродействию АЛУ и УУ процессора. Цикл обращения к СОП составляет 1-2 такта. Объем СОП очень небольшой. Во многих случаях СОП называют также *внутренней регистровой памятью* процессора. Регистры СОП используют для временного хранения результатов операции в АЛУ, операндов, служебных констант, очень коротких наборов команд обрабатываемой программы и т.д.

По своей сути СОП является буферной памятью, которая в какой-то степени сглаживает разрыв в быстродействии процессора и ОП. Однако ее незначительный объем не позволяет получить приемлемое решение проблемы, поэтому в процессе эволюции ЭВМ возник другой иерархический уровень буферной памяти, быстродействие которого несколько ниже СОП, а емкость существенно больше.

Кэш-память. Память этого типа является быстродействующим буфером достаточно большого объема между процессором (его внутренней памятью) и сравнительно медленно действующей ОП. Ее объем (одноуровневая кэш) составляет около 16-256 Кбайт на 4-8 Мбайт ОП. Эта память недоступна программисту (*cash* в переводе означает *тайник*). Кэш-память, как уже отмечалось, располагается в непосредственной близости от процессора, а кэш верхних уровней – непосредственно в кристалле процессора. В настоящее время кэш верхнего уровня и СОП стали фактически единым иерархическим уровнем внутренней памяти процессора. В IBM PC БИС нижнего уровня кэш располагается на процессорной шине. Информация в кэш-память закачивается из ОП небольшими блоками, при этом ненужные блоки удаляются из кэш обратно в ОП. Алгоритмы обмена кэш-памяти и ОП весьма строги и будут рассмотрены далее. Наличие кэш-памяти позволяет сгладить различие в быстродействии процессора и ОП. Кроме того, кэш-память дает возможность в ряде случаев не прерывать работу процессора при обмене внешних устройств с ОП в режиме прямого доступа (DMA).

*Внешняя память***.** Потребность в памяти, объем которой существенно превосходил бы размер существующих ОП, возникла в процессе эксплуатации уже первых ЭВМ. Такая память могла решить многие проблемы, связанные с вводом в ЭВМ больших программ, которые было невозможно разместить в ОП, и особенно с хранением больших наборов данных. Первоначально в качестве внешней памяти ЭВМ использовались накопители на магнитных барабанах (НМБ) и магнитных лентах (НМЛ). Затем были разработаны и созданы накопители на жестких и гибких магнитных дисках (НМД), которые стали интенсивно вытеснять более медленные НМЛ. Впоследствии были созданы накопители на оптических и магнитооптических дисках.

В настоящее время основным типом устройства внешней памяти является НМД. Внешнюю память на НМД иногда называют *оперативным* внешним запоминающим устройством (ВЗУ). НМЛ стали использоваться как *архивные* ВЗУ (стримеры), предназначенные для резервного хранения информации. К этому же классу ВЗУ относятся накопители на оптических и магнитооптических дисках. Все перечисленные ВЗУ имеют быстродействие во много раз меньше, чем ОП, и информация, хранимая на них, не может непосредственно перерабатываться процессором. Перед обработкой в процессоре информация с ВЗУ должна быть обязательно помещена в ОП. Емкость ВЗУ в ряде случаев для конкретной ЭВМ и конкретной задачи можно считать бесконечной.

Ниже рассматриваются принципы построения только внутренней памяти ЭВМ.

4.2. СПОСОБЫ ОРГАНИЗАЦИИ ПАМЯТИ

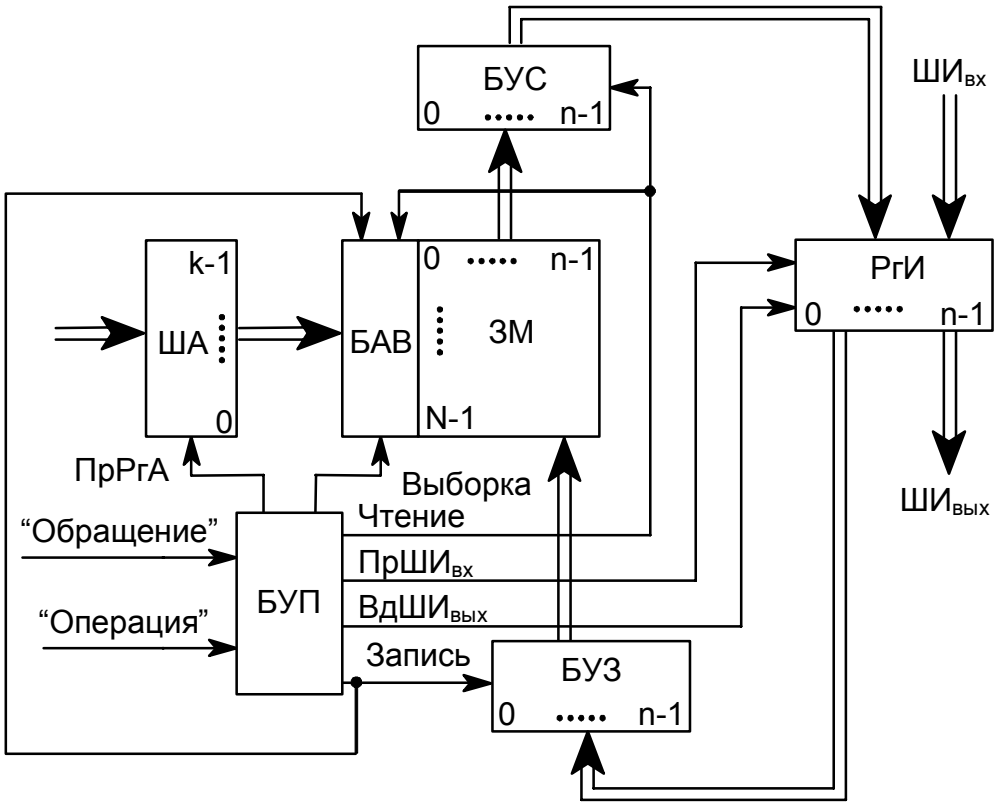
Функционально ЗУ любого типа всегда состоят из запоминающего массива, хранящего информацию, и вспомогательных, весьма сложных блоков, служащих для поиска в массиве, записи и считывания (и, если требуется, для регенерации).

Запоминающий массив (ЗМ) состоит из множества одинаковых запоминающих элементов (ЗЭ). Все ЗЭ организованы в ячейки, каждая из которых предназначена для хранения единицы информации в виде двоичного кода, число разрядов которого определяется шириной выборки. Способ организации памяти зависит от методов размещения и поиска информации в ЗМ. По этому признаку различают адресную, ассоциативную и стековую память.

4.2.1. АДРЕСНАЯ ПАМЯТЬ

В памяти с адресной организацией размещение и поиск информации в ЗМ основаны на использовании адреса хранения единицы информации, которую в дальнейшем для краткости будем называть *словом*. Адресом служит номер ячейки ЗМ, в которой это слово размещается. При записи (считывании) слова в ЗМ инициирующая эту операцию команда должна указывать адрес (номер) ячейки, по которому надо произвести запись (считывание).

На рис. 4.2 изображена обобщенная структура адресной памяти.



N – число ячеек памяти (n-разрядных);
РгА – регистр адреса;
РгИ – информационный регистр;
БАВ – блок адресной выборки (адресный дешифратор);
БУС – блок усилителей считывания;
БУЗ – блок усилителей записи;
БУП – блок управления памятью;
$ПрШИ_{вх}$ – прием с входной информационной шины;
$ВдШИ_{вых}$ – выдача на выходную информационную шину;
ША – шина адреса

Рис. 4.2. Обобщенная структура адресного ЗУ

Цикл обращения к памяти инициализируется поступающим в БУП сигналом "Обращение". Общая часть цикла обращения включает в себя прием в РгА с шины адреса (ША) адреса обращения и прием в БУП управляющего сигнала "Операция", указывающего вид запрашиваемой операции (считывание или запись).

*Считывание***.** БАВ дешифрирует адрес и посылает сигнал, выделяющий заданную адресом ячейку ЗМ. В общем случае БАВ может также посылать в выделенную ячейку памяти сигналы, настраивающие ЗЭ ячейки на запись или считывание. После этого записанное в ячейку слово считывается усилителями БУС и передается в РгИ. Затем в памяти с разрушающим считыванием происходит регенерация информации путем записи слова из РгИ через БУЗ в ту же ячейку ЗМ. Операция считывания завершается выдачей слова из РгИ на выходную информационную шину $ШИ_{вых}$.

Запись. Помимо указанной выше общей части цикла обращения происходит прием записываемого слова с входной шины $ШИ_{вх}$ в РгИ. Сама запись в общем случае состоит из двух операций – очистки ячейки и собственно записи. Для этого БАВ сначала производит выборку и очистку ячейки, заданной адресом в РгА. Очистка ячейки ЗМ (приведение в исходное состояние) может осуществляться по-разному. В частности, в памяти с разрушающим считыванием очистку можно производить сигналом считывания слова в ячейке при блокировке БУС (чтобы в РгИ не поступила информация). Затем в выбранную ячейку записывается новое слово.

Необходимость в операции очистки ячейки перед записью, так же как и в операции регенерации информации при считывании, определяется типом используемых ЗЭ, способами управления, особенностями электронной структуры БИС памяти, поэтому в полупроводниковых памятях эти операции могут отсутствовать.

БУП генерирует необходимые последовательности управляющих сигналов, инициирующих работу отдельных узлов памяти. Следует иметь в виду, что БУП может быть весьма сложным устройством (своеобразным управляющим контроллером, имеющим собственную кэш-память), придающим БИСу памяти в целом специальные потребительские свойства, такие как многопортовость, конвейерная выдача информации и т.п.

4.2.2. АССОЦИАТИВНАЯ ПАМЯТЬ

В памяти этого типа поиск информации происходит не по адресу, а по ее содержанию. Под содержанием информации в данном случае понимается не смысловая нагрузка лежащего на хранении в ячейке памяти слова, а содержание ЗЭ ячейки памяти, т.е. побитовый состав записанного двоичного слова. При этом ассоциативный запрос (признак) также представляет собой двоичный код с определенным побитовым составом. Поиск по ассоциативному признаку происходит параллельно во времени для всех ячеек ЗМ и представляет собой операцию сравнения содержимого разрядов регистра признака с содержимым соответствующих разрядов ячеек памяти. Для организации такого поиска все ЗЭ ЗМ снабжены однобитовыми процессорами, поэтому в ряде случаев память такого типа рассматривают как многопроцессорную систему.

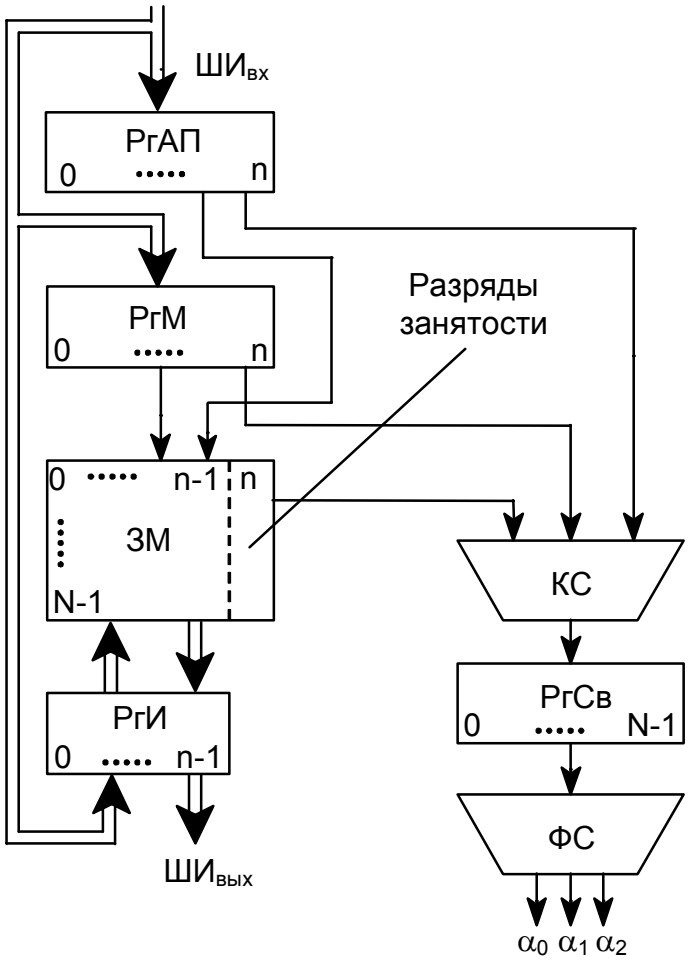
Полностью ассоциативная память большого объема является очень дорогостоящим устройством, поэтому для ее удешевления уменьшают число однобитовых процессоров до одного на ячейку памяти. В этом случае сравнение ассоциативного запроса с содержимым ячеек памяти идет последовательно для отдельных разрядов, параллельно во времени для всех ячеек ЗМ.

При очень больших объемах памяти на определенных классах задач ассоциативный поиск существенно ускоряет обработку данных и уменьшает вероятность сбоя в ЭВМ. Кроме того, ассоциативные ЗУ с блоками соответствующих комбинационных схем позволяют выполнить в памяти достаточно сложные логические опера-

ции: поиск максимального или минимального числа в массиве, поиск слов, заключенных в определенные границы, сортировку массива и т.д.

Следует отметить, что ассоциативный поиск можно реализовать и в компьютере с обычной адресной памятью, последовательно вызывая записанные в ячейки памяти слова в процессор и сравнивая их с некоторым ассоциативным признаком (шаблоном). Однако при больших объемах памяти на это будет затрачено много времени. При использовании ассоциативной памяти можно, не считывая слов из ОП в процессор, за одно обращение определить количество слов, отвечающих тому или иному ассоциативному запросу. Это позволяет в больших базах данных очень оперативно реализовать запрос типа: сколько жителей области не представило декларацию о доходах и т.п.

В некоторых специализированных ЭВМ ОП или его часть строится таким образом, что позволяет реализовать как ассоциативный, так и адресный поиск информации.



N – число ячеек памяти (n-разрядных);
РгАП – регистр ассоциативного признака;
РгМ – регистр маски;
РгИ – информационный регистр;
РгСв – регистр совпадения;
ФС – формирующая схема;
КС – комбинационная схема;
$ШИ_{вх}$ – входная информационная шина;
$ШИ_{вых}$ – выходная информационная шина;
ЗМ – запоминающий массив

Рис. 4.3. Обобщенная структура ассоциативного ЗУ

Упрощенная структурная схема ассоциативной памяти, в которой все ЗЭ ЗМ снабжены однобитовыми процессорами, приведена на рис. 4.3.

Первоначально рассмотрим операцию, называющуюся *контроль ассоциации*. Эта операция является общей для операции считывания и записи, а также имеет самостоятельное значение.

По входной информационной шине в РгАП поступает n-разрядный ассоциативный запрос, т.е. заполняются разряды от 0 до n-1. Одновременно в РгМ поступает код маски поиска, при этом n-й разряд РгМ устанавливается в 0. Ассоциативный поиск производится лишь для совокупности разрядов РгАП, которым соответствуют 1 в РгМ (незамаскированные разряды РгАП). Для слов, в которых цифры в разрядах совпали с незамаскированными разрядами РгАП, КС устанавливает 1 в соответствующие разряды РгСв и 0 в остальные разряды.

Комбинационная схема формирования результата ассоциативного обращения ФС формирует из слова, образовавшегося в РгСв, как минимум три сигнала:

- α_0 – отсутствие в ЗМ слов, удовлетворяющих ассоциативному признаку;
- α_1 – наличие одного такого слова;
- α_2 – наличие более чем одного слова.

Возможны и другие операции над содержимым РгСв, например подсчет количества единиц, т.е. подсчет слов в памяти, удовлетворяющих ассоциативному запросу, и т.п.

Формирование содержимого РгСв и α_0, α_1, α_2 по содержимому РгАП, РгМ, ЗМ и называется операцией контроля ассоциации.

Считывание. Сначала производится контроль ассоциации по признаку в РгАП. Затем:

- $\alpha_0 = 1$ – считывание отменяется из-за отсутствия искомой информации;
- $\alpha_1 = 1$ – считывается в РгИ найденное слово, после чего выдается на $ШИ_{вых}$;
- $\alpha_2 = 1$ – считывается слово, имеющее, например, наименьший номер среди ячеек, отмеченных 1 в РгСв, после чего выдается на $ШИ_{вых}$.

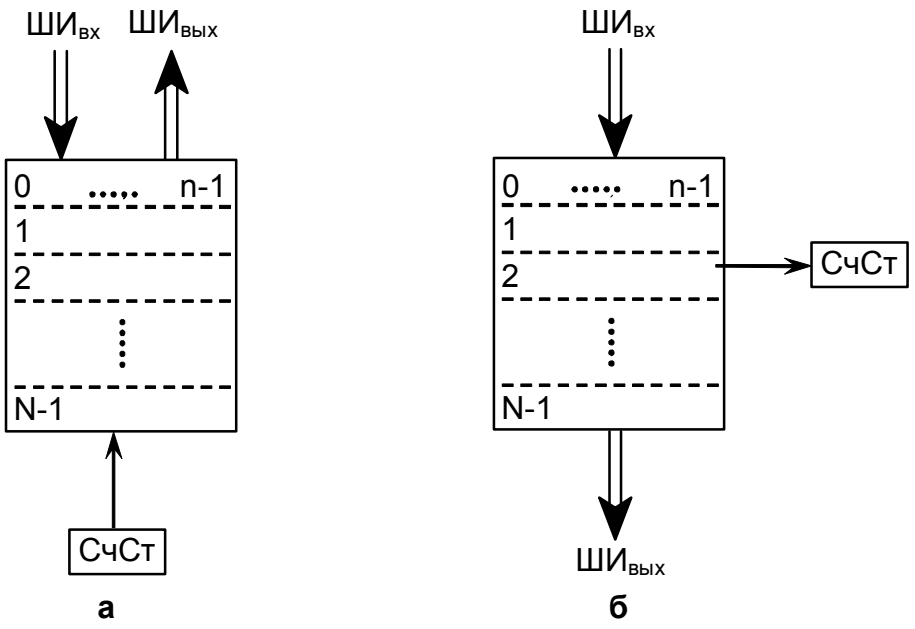
Запись. Сначала отыскивается свободная ячейка (полагаем, что в разряде занятости свободной ячейки записан 0). Для этого выполняется контроль ассоциации при РгАП=111...10 и РгМ=000...01, т.е. n-й разряд РгАП устанавливается в 0, а n-й разряд РгМ – в 1. При этом свободная ячейка отмечается 1 в РгСв. Для записи выбирают свободную ячейку, например, с наименьшим номером. В нее записывается слово, поступившее с $ШИ_{вх}$ в РгИ.

Следует отметить, что на данной схеме не изображены блоки БУП, БУС, БУЗ, которые есть в реальных устройствах. Кроме того, для построения ассоциативной памяти требуются запоминающие элементы, допускающие считывание без разрушения.

4.2.3. СТЕКОВАЯ ПАМЯТЬ (МАГАЗИННАЯ)

Стековая память, так же как и ассоциативная, является безадресной. Стековая память может быть организована как аппаратно, так и на обычном массиве адресной памяти.

В случае аппаратной реализации ячейки стековой памяти образуют одномерный массив, в котором соседние ячейки связаны друг с другом разрядными цепями передачи слов (рис. 4.4). При этом возможны два типа устройств (а, б), принципы функционирования которых различны. Рассмотрим первоначально структуру на рис. 4.4, а.



$ШИ_{вых}$ – выходная информационная шина;
$ШИ_{вх}$ – входная информационная шина;
N – число ячеек памяти (n-разрядных);
СчСт – счетчик стека

Рис. 4.4. Обобщенная структура аппаратного стека: а – стек LIFO; б – стек FIFO

Запись нового слова, поступившего с $ШИ_{вх}$, производится в верхнюю (нулевую) ячейку, при этом все ранее записанные слова (включая слово в ячейке 0) сдвигаются вниз, в соседние ячейки, номера которых на единицу больше. Считывание возможно только из верхней (нулевой) ячейки памяти. Основной режим – это считывание с удалением. При этом все остальные слова в памяти сдвигаются вверх, в соседние ячейки с меньшими номерами. В такой памяти реализуется правило: *последний пришел – первый ушел*. Стеки подобного типа принято называть стеками LIFO (Last In – First Out).

В ряде случаев устройства стековой памяти предусматривают также операцию простого считывания слова из ячейки 0 без его удаления и сдвига остальных слов. При использовании стека для запоминания параметров инициализации контроллеров каких-либо устройств ЭВМ обычно предусматривается возможность считывания содержимого любой ячейки стека без его удаления, т.е. считывание содержимого не только ячейки 0.

О первом слове, посылаемом в стек, говорят, что оно располагается на *дне стека*. О последнем посланном (по времени) в стек слове говорят, что оно находится в *вершине стека*. Таким образом, ячейка N-1 – дно стека, а ячейка 0 – вершина.

Обычно аппаратный стек снабжается счетчиком стека СчСт, показывающим общее количество занесенных в память слов (СчСт = 0 – стек пустой). При заполнении стека полностью он запрещает дальнейшие операции записи.

Стековый принцип организации памяти можно реализовать не только в специально предназначенных для этого устройствах. Стековая организация данных возможна и на обычной адресной памяти с произвольным обращением (программный стек). Для организации стека LIFO в этом случае необходима еще одна ячейка памяти (регистр), в которой всегда хранится адрес вершины стека и которая называется *указателем стека*. Обычно в качестве указателя стека используют один из внутренних регистров процессора. Кроме этого, требуется соответствующее програм-

мное обеспечение. Принципы стековой организации данных на обычной адресной памяти иллюстрируются схемой на рис. 4.5.

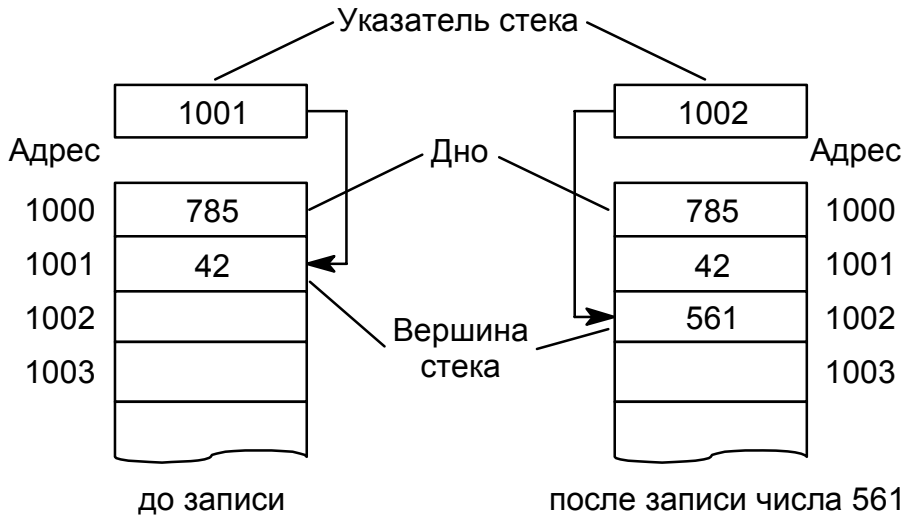


Рис. 4.5. Запись слова в программный стек LIFO

В отличие от аппаратного стека данные, размещенные в программном стеке, при записи нового числа или считывании не перемещаются. Запись каждого нового слова осуществляется в ячейку памяти, следующую по порядку за той, адрес которой содержится в указателе стека. После записи нового слова содержимое указателя стека увеличивается на единицу (см. рис. 4.5). Таким образом, в программном стеке перемещаются не данные, а вершина стека. При считывании слова из стека происходит обратный процесс. Слово считывается из ячейки, адрес которой находится в указателе стека, после чего содержимое указателя стека уменьшается на единицу.

Если вновь загружаемые в стек слова размещаются в ячейках памяти с последовательно увеличивающимися адресами, стек называют *прямым.* Если адреса последовательно убывают, то – *перевернутым.* В большинстве случаев используется перевернутый стек, что связано с особенностями аппаратной реализации счетчиков внутри процессора.

Чем удобна такая форма организации памяти? Забегая вперед, можно отметить, что любая команда, выполняемая в процессоре, в общем случае должна содержать код операции (КОП), адрес первого и второго операндов и адрес занесения результата. Для экономии памяти и сокращения времени выполнения машинной команды процессором желательно уменьшить длину команды. Пределом такого уменьшения является длина безадресной команды, т.е. просто КОП. Именно такие команды оказываются возможными при стековой организации памяти, так как при правильном расположении операндов в стеке достаточно последовательно их извлекать и выполнять над ними соответствующие операции.

Помимо рассмотренной выше стековой памяти типа LIFO в ЭВМ используются стековые памяти другого типа, реализующие правило: *первый пришел – первый ушел*. Стеки подобного типа принято называть стеками FIFO (First In – First Out). Такая стековая память широко используется для организации различного рода очередей (команд, данных, запросов и т.д.). Обобщенная структура аппаратного стека типа FIFO представлена на рис. 4.4, б.

Как и в предыдущем случае, ячейки стековой памяти образуют одномерный массив, в котором соседние ячейки связаны друг с другом разрядными цепями передачи слов. Запись нового слова, поступившего с $ШИ_{вх}$, осуществляется в верхнюю (нулевую) ячейку, после чего оно сразу перемещается вниз и записывается в по-

следнюю по счету незаполненную ячейку. Если стек перед записью был пустой, слово сразу попадает в ячейку с номером N-1, т.е. на дно стека. Считывание возможно только из нижней ячейки с номером N-1 (дно стека). Основной режим – это считывание с удалением. При этом все последующие (записанные) слова сдвигаются вниз, в соседние ячейки, номера которых на единицу больше. При заполнении стека счетчик (СчСт) запрещает дальнейшие операции записи в стек.

Таким образом, в отличие от стека LIFO, в стеке FIFO перемещается не дно, а вершина. Записываемые в стек FIFO слова постепенно продвигаются от вершины ко дну, откуда и считываются по мере необходимости, причем темп записи и считывания определяются внешними управляющими сигналами и не связаны друг с другом.

Программная реализация стека FIFO в настоящем разделе не рассматривается, поскольку на практике используется достаточно редко.

4.3. СТРУКТУРЫ АДРЕСНЫХ ЗУ

Адресные ЗУ наиболее широко используются в современных ЭВМ для построения самых разнообразных устройств памяти. В процессе эволюции ЭВМ принципы построения и аппаратная реализация данных ЗУ прошли очень большой путь развития от первых ЗУ на электромагнитных реле до современных БИСов памяти емкостью в сотни Мбайт, которые в качестве ЗЭ используют либо разнообразные триггерные схемы на биполярных полупроводниках, либо МОП-структуры. При этом тип используемых ЗЭ влияет на структуру ЗУ. Кроме того, структура ЗУ во многом определяется особенностями его применения в конкретных устройствах ЭВМ. Все это привело к тому, что в процессе развития возникло весьма большое разнообразие структур ЗУ, которые различаются по способу организации, быстродействию, объему, аппаратурным затратам, стоимости.

Ранее отмечалось, что основной частью любой памяти является запоминающий массив (ЗМ), представляющий собой совокупность ЗЭ, соединенных определенным образом. ЗМ называют еще запоминающей матрицей. Каждый ЗЭ хранит бит информации и должен реализовывать следующие режимы работы:

- хранение состояния (0 или 1);
- выдачу сигнала состояния (считывание);
- запись информации (0 или 1).

К ЗЭ должны поступать управляющие сигналы для задания режима работы, а также сигналы при записи. При считывании ЗЭ должен выдавать сигнал о своем состоянии, поэтому любой ЗМ имеет систему *адресных* и *разрядных* линий (проводников).

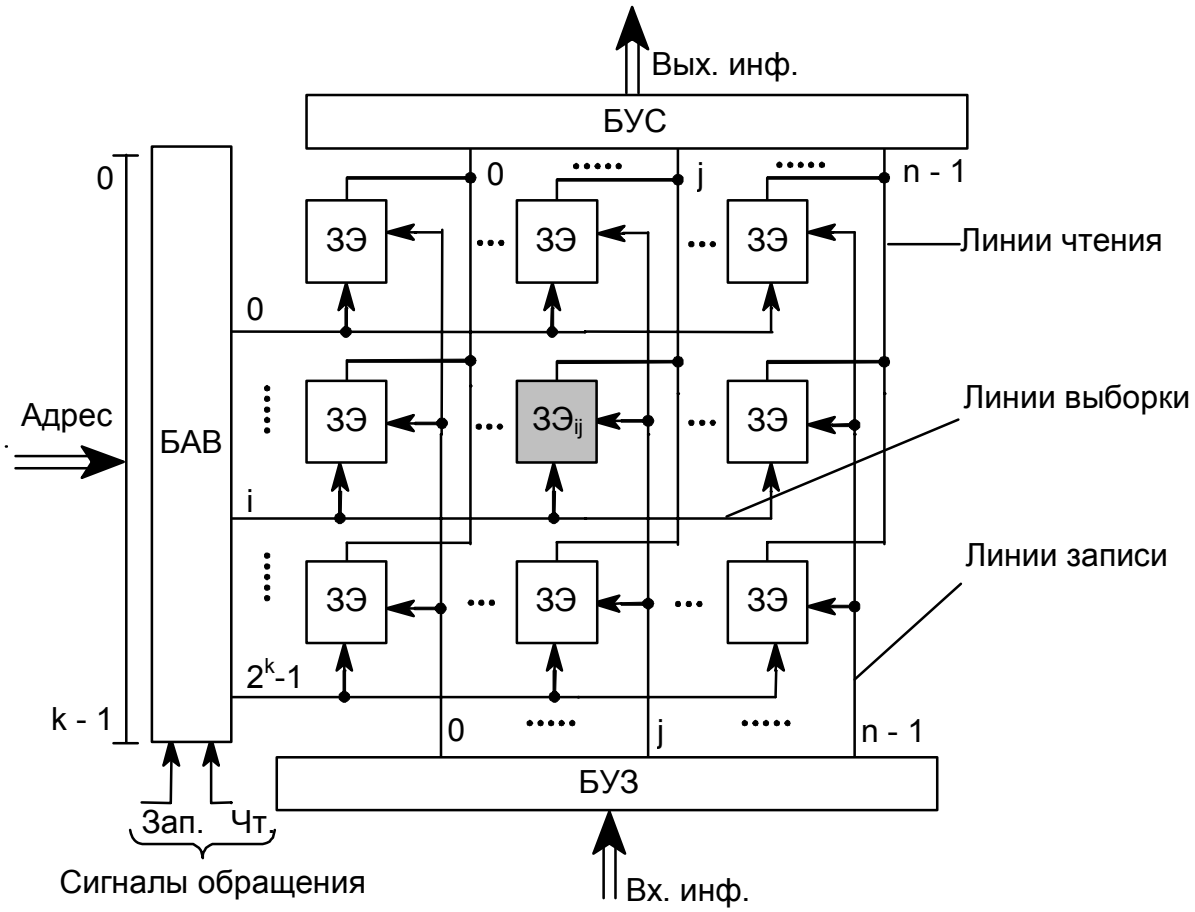
Адресные линии используются для выделения по адресу совокупности ЗЭ, которым устанавливается режим считывания или записи. Число ЗЭ, входящих в эту совокупность, равно ширине выборки. Иными словами, с помощью адресных линий происходит выбор необходимой ячейки памяти. Разрядные линии используются для записи или считывания информации в ЗЭ каждого разряда ячейки памяти.

Адресные и разрядные линии носят общее название *линий выборки*. В зависимости от числа таких линий, соединенных с одним ЗЭ, различают двух-, трехкоординатные ЗУ и т.д., называемые ЗУ типа 2D, 3D, 2.5D, 2D-M (от слова dimension – размерность), и их разнообразные модификации.

4.3.1. ЗУ ТИПА 2D

Организация ЗУ типа 2D обеспечивает двухкоординатную выборку каждого ЗЭ ячейки памяти. Основу ЗУ составляет плоская матрица из ЗЭ, сгруппированных в 2^k ячеек по n разрядов. Обращение к ячейке задается k-разрядным адресом, что дает одну координату. Выделение разрядов производится разрядными линиями записи и

считывания, что дает вторую координату. Очень упрощенная структура ЗУ типа 2D представлена на рис. 4.6.



БАВ - блок адресной выборки (адресный формирователь);
БУС – блок усилителей считывания;
БУЗ – блок усилителя записи;
ЗЭ – запоминающий элемент

Рис. 4.6. Структура ЗУ типа 2D

Адрес из k разрядов поступает на блок адресной выборки БАВ (который называют также адресным формирователем), управляемый сигналами Чт и Зап. Основу БАВ составляет дешифратор с 2^k выходами, который при поступлении на его вход адреса формирует сигнал для выбора линии i. В зависимости от сигнала Чт или Зап БАВ в общем случае выдает сигнал, настраивающий ЗЭ i-й ячейки (i-й линии) на чтение либо на запись. Выделение разряда j в i-м слове (ЗУ серого цвета) производится второй координатной линией. При записи по линии j от БУЗ поступает сигнал, устанавливающий выбранный для записи $ЗЭ_{i,j}$ в состояние 0 или 1. При считывании на БУС по линии j поступает сигнал о состоянии $ЗЭ_{i,j}$.

Следует иметь в виду, что ЗЭ должны допускать объединение выходов для работы на общую линию с передачей сигналов только от выбранного ЗЭ. Это свойство ЗЭ используется во всех современных ЗУ.

Таким образом, каждая адресная линия выборки ячейки памяти в общем случае передает три сигнала:

- выборка при записи;
- выборка при считывании;
- отсутствие выборки.

Однако во многих современных ЗУ достаточно только двух сигналов – выборка и отсутствие выборки.

Каждая разрядная линия записи передает в ЗЭ записываемый бит информации, а разрядная линия считывания – считываемый из ЗЭ бит информации. Линии записи и считывания могут быть объединены в одну при использовании ЗЭ, допускающих соединение выхода со входом записи. В современных ЗУ широко используются совмещенные функции линий считывания и записи.

ЗУ типа 2D являются быстродействующими и достаточно удобными для реализации. Однако такие ЗУ неэкономичны по объему оборудования из-за наличия дешифратора на 2^k выходов. В настоящее время структуры типа 2D используются, в основном, в ЗУ небольшой емкости (не более 1 К).

4.3.2. ЗУ ТИПА 3D

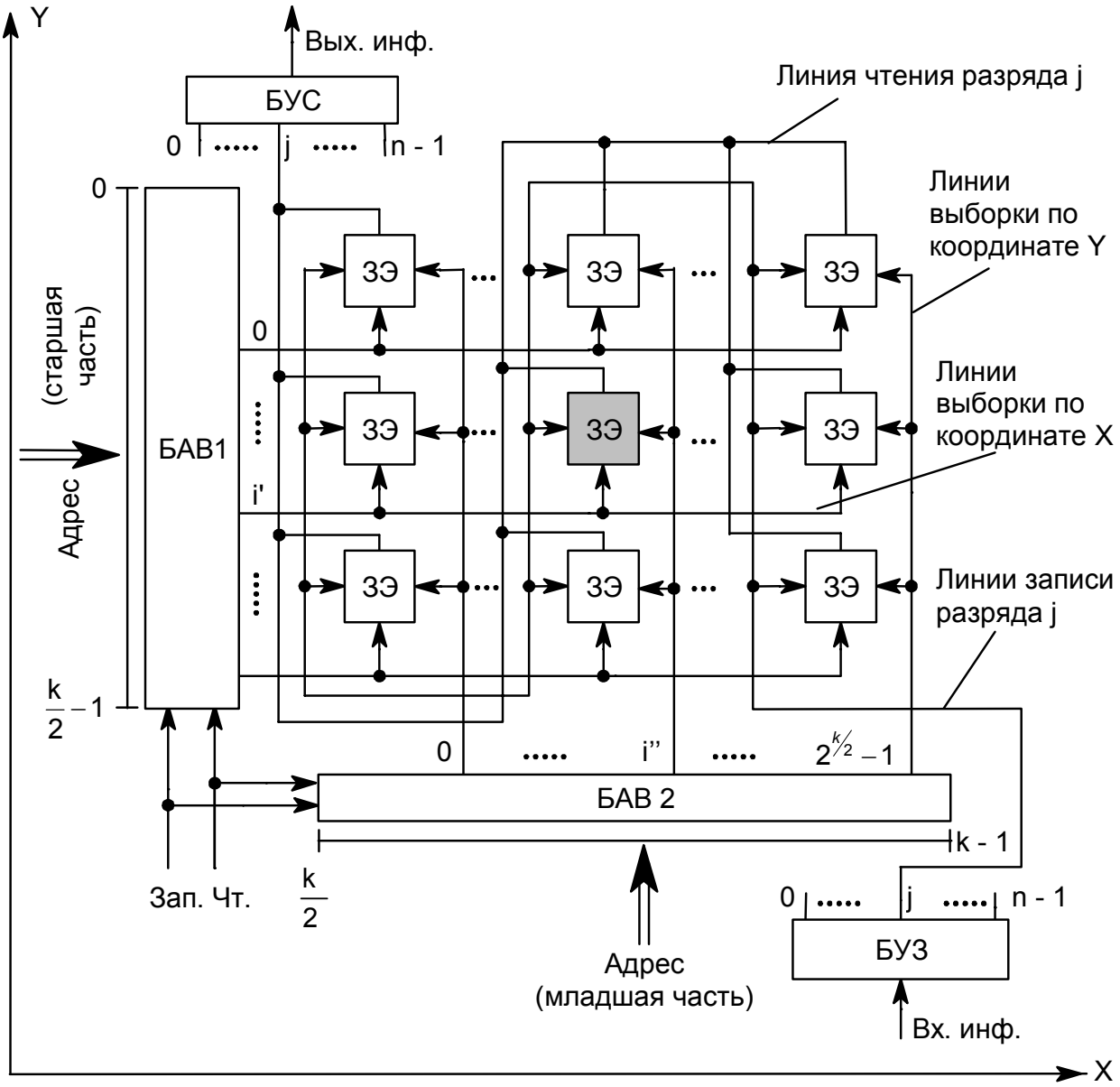
Для построения ЗУ больших объемов используют другую схему и другие типы ЗЭ, которые имеют не один, а два конъюнктивно связанных входа выборки. В этом случае адресная выборка осуществляется только при одновременном появлении двух сигналов. Использование таких ЗЭ позволяет строить ЗУ с трехкоординатным выделением ЗЭ. Итак, ЗУ типа 3D отличается от 2D тем, что к каждому ЗЭ подходят три линии выборки: две координатные и одна разрядная.

Запоминающий массив ЗУ типа 3D представляет собой пространственную матрицу, составленную из n плоских матриц. Каждая плоская матрица представляет собой ЗМ для запоминания j-х разрядов всех слов, т.е. запоминающие элементы для одноименных разрядов всех хранимых в ЗУ чисел сгруппированы в квадратную матрицу из $2^{k/2}$ рядов по $2^{k/2}$ ЗЭ в каждом. Это означает, что к записи или считыванию готов только тот элемент, для которого сигналы адресной выборки по координатам X и Y совпали. Для адресной выборки ЗЭ в плоской матрице необходимо задать две его координаты в ЗМ.

Структура матрицы j-го разряда в ЗУ типа 3D представлена на рис. 4.7. Код адреса i-й ячейки памяти разделяется на старшую и младшую части, каждая из которых поступает на свой адресный формирователь. Адресный формирователь БАВ1 выдает сигнал выборки на линию i', а БАВ2 – на линию i''. В результате в ЗМ оказывается выбранным ЗЭ, находящийся на пересечении этих линий (двух координат), т.е. адресуемый кодом i=i' / i'' (ЗЭ серого цвета). Адресные формирователи управляются сигналами Чт и Зап и в зависимости от них выдают сигналы выборки для считывания или для записи. При считывании сигнал о состоянии выбранного ЗЭ поступает по j-линии считывания к БУС (третья координата ЗЭ при считывании). При записи в выбранный ЗЭ будет занесен 0 или 1 в зависимости от сигнала записи в j-й разряд, поступающего по j-й линии от БУЗ (третья координата ЗЭ при записи). Для полупроводниковых ЗУ характерно объединение в одну линию разрядных линий записи и считывания.

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

Запоминающие устройства типа 3D более экономичны по оборудованию, чем ЗУ типа 2D. Действительно, сложность адресного формирователя с m входами пропорциональна 2^m, отсюда сложность двух адресных формирователей ЗУ типа 3D, пропорциональная $2 \cdot 2^{k/2}$, значительно меньше сложности адресного формирователя ЗУ типа 2D, пропорциональной 2^k. Поэтому структура типа 3D позволяет строить ЗУ большего объема, чем структура 2D.



БАВ - блок адресной выборки (адресный формирователь);
БУС – блок усилителей считывания;
БУЗ – блок усилителя записи;
ЗЭ – запоминающий элемент

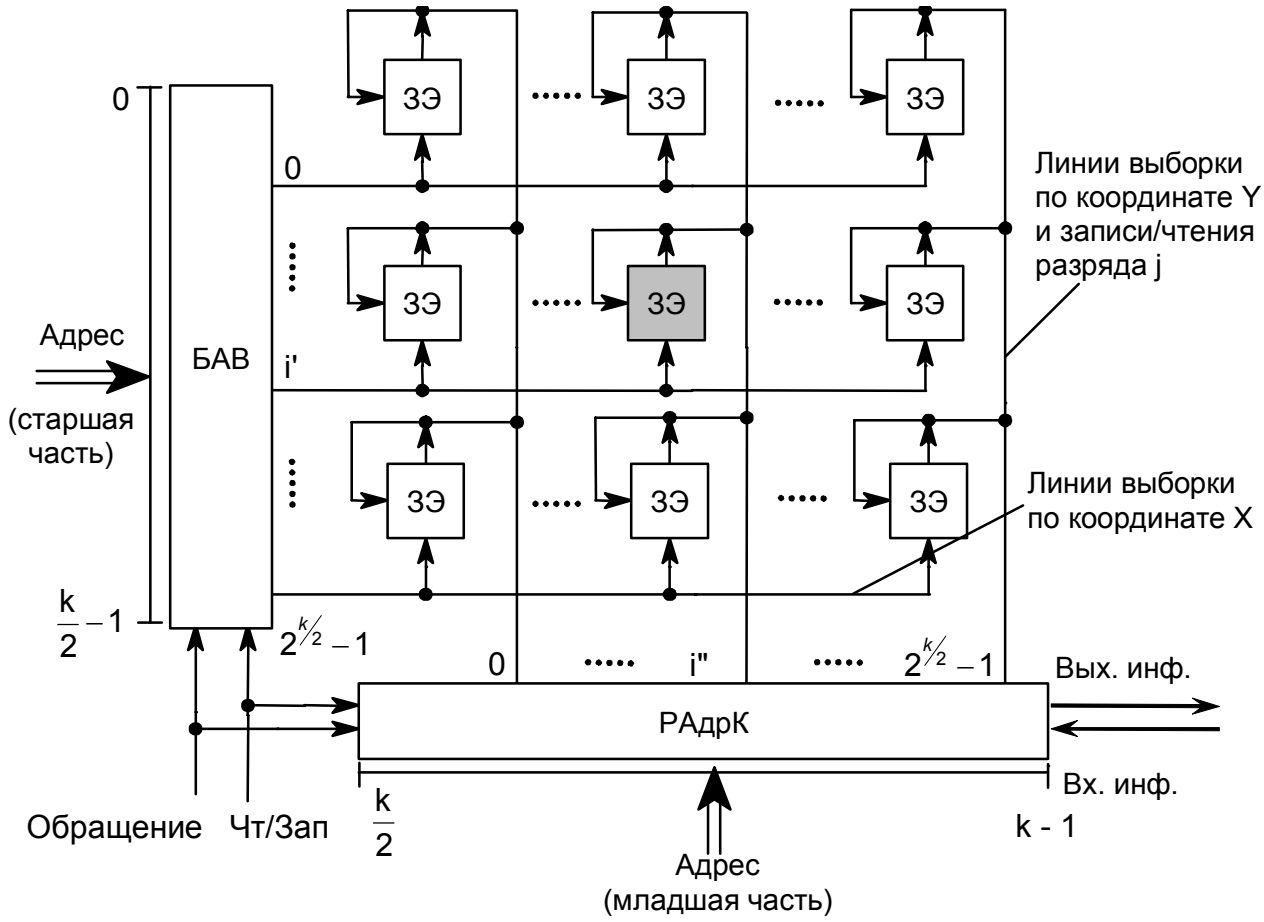
Рис. 4.7. Структура ЗУ типа 3D для j-го разряда

Структура типа 3D является наиболее удобной для построения статических ЗУ на многоэмиттерных биполярных транзисторах.

4.3.3. ЗУ ТИПА 2D-M

В ЗУ типа 2D-M ЗМ для записи n-разрядных двоичных чисел состоит из n плоских матриц для одноименных разрядов всех чисел, что имеет место в ЗУ типа 3D. Однако процесс записи и считывания информации существенно отличается, поскольку в ЗУ типа 2D-M используются другие ЗЭ, к каждому из которых подходят только две координатные линии. ЗЭ таких ЗУ имеют два входа (координатный и записи) и один выход, но обычно их выход объединен со входом записи.

Структура одноразрядного ЗУ типа 2D-M (ЗМ для j-го разряда всех ячеек памяти) приведена на рис. 4.8.



БАВ - блок адресной выборки (адресный формирователь);
РАдрК – разрядно - адресный коммутатор;
ЗЭ – запоминающий элемент

Рис. 4.8. Структура ЗУ типа 2D-M для j-го разряда

Как и в ЗУ типа 3D, код адреса i-й ячейки памяти разделяется на две части, одна из которых поступает на БАВ, а другая – на разрядно-адресный коммутатор (РАдрК). РАдрК является не только устройством адресной выборки j-го разряда i-й ячейки памяти, но также устройством записи и считывания информации, хранимой в ЗЭ. Если на БАВ и РАдрК не приходит сигнал обращения к памяти Обр, то на их выходных линиях не возникают действующие на ЗЭ сигналы и все ЗЭ находятся в режиме хранения. При наличии сигнала Обр выполняется считывание или запись в зависимости от сигнала *Чт/Зап*. При считывании БАВ выдает по линии i′ сигнал выборки для считывания, по которому со всех ЗЭ линии i′ сигналы их состояний поступают на РАдрК. Коммутатор РАдрК мультиплексирует эти сигналы и передает на выход (*Вых. инф*.) сигнал только с линии i″. При записи БАВ выдает по линии i′ сигнал выборки для записи. Коммутатор РАдрК в зависимости от значения сигнала *Вх. инф*. выдает сигнал записи 0 или 1 на линию i″ и сигналы, не воздействующие на ЗЭ, в остальные линии. В результате запись производится только в ЗЭ, лежащий на пересечении координатных линий i′и i″, причем i′/i″ = i (ЗЭ серого цвета).

Структура типа 2D-M является наиболее удобной для построения полупроводниковых ЗУ на МОП-структурах и широко используется в настоящее время как в динамических оперативных, так и в постоянных ЗУ.

Структура ЗУ типа 2,5D специально не рассматривается. По схеме 2,5D строят ЗУ на магнитных сердечниках. Некоторые особенности организации ЗУ такого типа более подробно рассмотрены в п. 4.4.1.

4.4. ЭЛЕМЕНТЫ ЗУ С ПРОИЗВОЛЬНЫМ ОБРАЩЕНИЕМ

Непрерывное совершенствование элементной базы, а также многообразие вариантов целевого использования ЗУ внутренней памяти ЭВМ привело к созданию большого количества разновидностей ЗЭ. Ниже кратко рассмотрены наиболее характерные типы ЗЭ, используемые в ЗУ универсальных ЭВМ и специализированных цифровых устройств.

4.4.1. ЗЭ НА ФЕРРИТОВЫХ КОЛЬЦАХ

Памяти на магнитных (ферритовых) сердечниках с прямоугольной петлей гистерезиса появились в начале 50-х годов и сыграли большую роль в увеличении объемов ОП и производительности ЭВМ. Однако появившиеся ЗУ на дискретных биполярных транзисторах, а в середине 60-х годов первые БИСы памяти малой степени интеграции стремительно вытеснили ЗУ на магнитных сердечниках. В настоящее время ЗУ на магнитных сердечниках используются только в специализированных ЭВМ, работающих в особых условиях. Это связано с тем, что такие ЗУ энергонезависимы, не боятся радиации и сильных температурных перепадов.

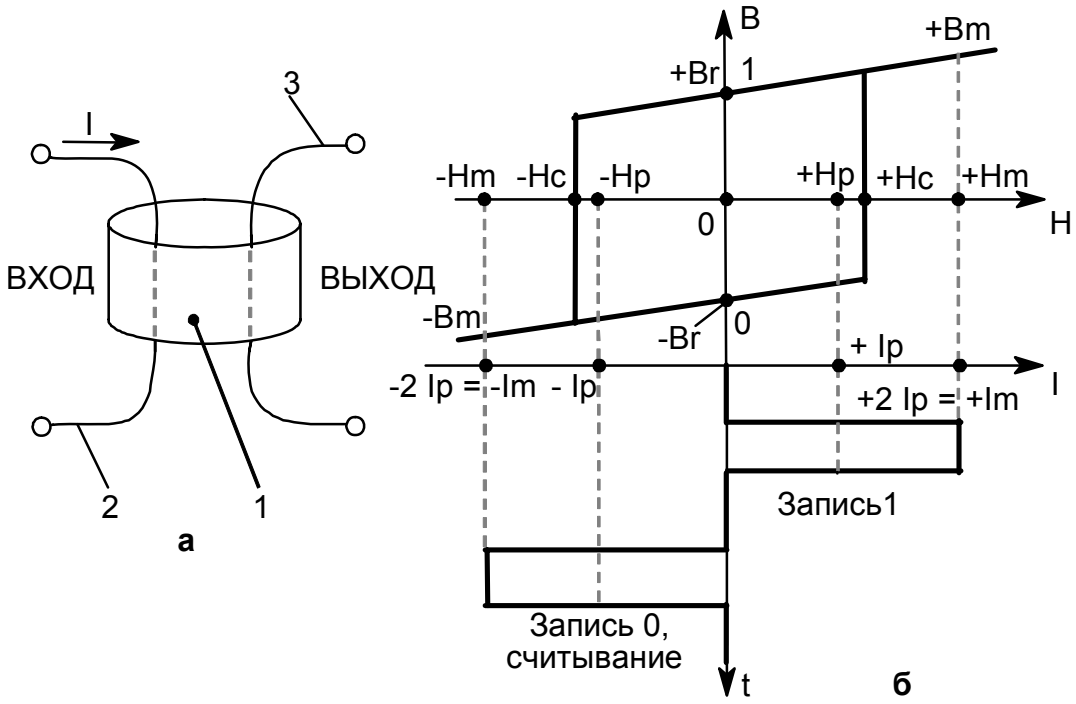
Следует отметить, что физические принципы записи информации на магнитный сердечник лежат в основе принципов записи на современные магнитные носители.

Принцип действия ферритового сердечника как запоминающего элемента поясняет рис. 4.9, на котором изображены сердечник с входной и выходной одновитковыми обмотками и его кривая намагничивания (петля гистерезиса), имеющая форму, близкую к прямоугольной.

Запоминание двоичных кодов основано на наличии у сердечника двух устойчивых состояний остаточного намагничивания противоположного знака. Состоянию, характеризующемуся остаточной индукцией $+B_r$, можно приписать значение 1, а состоянию с индукцией $-B_r$ – значение 0. Запись 1 в сердечник может быть произведена подачей в его входную обмотку импульса тока, создающего поле $+H_m$, превышающее коэрцитивную силу сердечника $+H_c$, при этом сердечник после снятия поля оказывается в состоянии $+B_r$. Для записи 0 во входную обмотку подается импульс тока, создающий поле $-H_m$, после снятия которого сердечник оказывается в состоянии $-B_r$. Для считывания во входную обмотку подается импульс тока той же полярности, как при записи 0, создающий поле, превышающее (по модулю) коэрцитивную силу H_c сердечника, при этом сердечник устанавливается в состояние 0. Если перед этим он находился в состоянии 1, то при считывании индукция в сердечнике изменяется от $+B_r$ до $-B_m$ и в выходной обмотке индуцируется ЭДС E_1 – сигнал считывания 1. Если сердечник находился в состоянии 0, то при считывании индукция в сердечнике меняется незначительно и индуцируемая при этом ЭДС E_0 (сигнал считывания 0) близка к нулю. Считывание сопровождается стиранием информации, записанной в сердечнике, поэтому при необходимости сохранения ранее хранившейся информации должна производится регенерация, восстанавливающая информацию в сердечнике.

Ферритовые ЗУ, использующиеся в настоящее время, имеют структуру 2,5D (в ЗУ этого типа запись сходна с записью в ЗУ типа 2D-M, а считывание осуществляется так же, как в ЗУ типа 3D). Как и в ЗУ типа 3D, запоминающий массив 2,5D для записи n-разрядных двоичных чисел состоит из n плоских матриц для одноименных разрядов всех чисел. Каждая такая матрица имеет три координатные линии, причем при считывании используются все три линии, а при записи только две. Таким образом, через каждый сердечник разрядной матрицы проходят одна горизонтальная и

одна вертикальная линии (соответственно от адресного и разрядно-адресного формирователей). Кроме того, все сердечники разрядной матрицы пронизываются общей обмоткой считывания, с которой снимаются выходные сигналы.



1 – сердечник; 2 – обмотка записи; 3 – обмотка считывания

Рис. 4.9. а - запоминающий элемент на ферритовом сердечнике;
б - петля гистерезиса сердечника

При считывании состояния сердечника в его координатные линии подаются импульсы тока $-I_p$. Ток $-I_p$ создает поле $-H_p$ такое, что $|-H_p| < |-H_c|$ и $2|-H_p| > |-H_c|$. На выбираемый сердечник воздействует поле двух токов $-I_p$, и он принимает состояние $-B_r$, генерируя в выходной обмотке сигнал 1, если сердечник ранее находился в состоянии $+B_r$, или сигнал 0, если сердечник находился в состоянии $-B_r$. Сердечники, воспринимающие ток $-I_p$ только по одной координатной линии, сохраняют свое состояние.

Для того, чтобы считанная информация сохранилась в ЗУ, сразу после окончания считывания выполняется регенерация – запись считанной информации из информационного регистра РгИ обратно в ЗМ.

Записи в ячейку ферритового ЗУ типа 2,5D всегда предшествует установка в состояние 0 всех сердечников ячейки, для чего выполняется операция считывания (без регенерации). При записи по адресной линии выбираемого сердечника поступает импульс тока $+I_p$. По разрядно-адресной линии поступает импульс тока $+I_p$ только при записи 1. Сердечник, в котором поля от двух импульсов $+I_p$ складываются, перемагничивается в состояние $+B_r$. Все остальные сердечники разрядной матрицы сохраняют свое прежнее состояние.

4.4.2. ЗЭ НА ПОЛУПРОВОДНИКОВЫХ ЭЛЕМЕНТАХ

Абсолютное большинство ЗУ внутренней памяти современных ЭВМ (а в универсальных ЭВМ общего назначения – 100%) построено на полупроводниковых ЗЭ. По сравнению с другими типами ЗЭ полупроводниковые ЗЭ имеют ряд существенных преимуществ. Основными преимуществами являются большее быстродействие,

компактность, меньшая стоимость, совместимость по сигналам с логическими схемами, технологичность.

По типу ЗЭ различают *биполярные ЗУ* с ЗЭ, построенными на биполярных транзисторах (по ТТЛ - или ЭСЛ – схемам), и *МОП-ЗУ* с ЗЭ, построенными на МОП-структурах.

Оба типа ЗУ широко используются, но имеют свои преимущества и недостатки. Биполярные ЗУ более быстродействующие и хорошо стыкуются с ТТЛ – и ЭСЛ –логикой. Но в настоящее время они еще довольно дороги и используются главным образом в качестве быстродействующих памятей, таких как управляющая память, СОП, кэш. Кроме того, такие ЗУ потребляют много энергии и имеют невысокую плотность упаковки элементов в кристалле. Запоминание информации в биполярных ЗУ происходит в триггерных ячейках, построенных на многоэмиттерных транзисторах. Это *статические* ЗУ, поскольку при включенном питании информация хранится в них любое время без регенерации.

МОП-ЗУ бывают как *статическими*, так и *динамическими*. В первом случае они построены на ЗЭ в виде триггеров. Во втором случае хранение информации основано на заряде "запоминающих емкостей", в качестве которых используются емкости некоторых цепей схемы. Это либо паразитная емкость затвора МОП-транзистора, либо специально сформированная емкость сток МОП-транзистора – подложка. Поскольку указанные емкости имеют ток утечки, то информацию в таких ЗУ необходимо регенерировать примерно через каждые 2 мс (операция называется *рефреш*). МОП-ЗУ сравнительно дешевы, потребляют мало энергии, имеют очень высокую плотность элементов на кристалле и, следовательно, большие емкости ЗУ в одном корпусе микросхемы. В настоящее время МОП-ЗУ широко используются для построения основной (оперативной) памяти ЭВМ различных классов. МОП-ЗУ менее быстродействующие, чем биполярные ЗУ.

Рассмотрим более подробно структуру полупроводникового ЗЭ на биполярных транзисторах и ЗЭ на МОП-структурах, в котором информация сохраняется в паразитной емкости затвора.

Биполярный ЗЭ

Простейший вариант биполярного ЗЭ представляет собой триггер на двух многоэмиттерных транзисторах с непосредственными связями, структура которого приведена на рис. 4.10. Запоминающие устройства на ЗЭ такого типа строятся по схеме 3D.

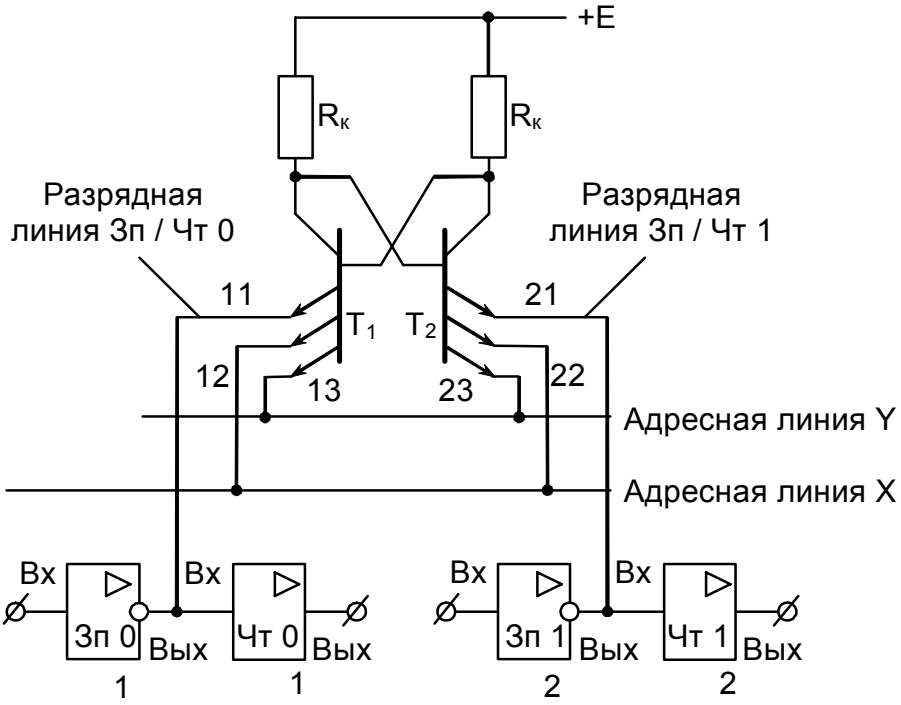
Эмиттеры 11 и 21 являются парафазными информационными входами ЗЭ и служат для записи в триггер 1 или 0. Эти же эмиттеры используются как выходы при считывании информации. Адресные эмиттеры 12, 22, 13, 23 образуют два конъюнктивно связанных входа выборки.

В режиме хранения (ЗЭ не выбран) эмиттерный ток открытого транзистора замыкается на землю через адресные эмиттеры и адресные линии (по крайней мере, через одну из линий), находящиеся под потенциалом логического 0 ($\leq$ 0.4 В). На входы усилителей записи подается такой уровень сигнала, чтобы на выходах невозбужденных усилителей записи напряжение было порядка 1-1.5 В, т.е. больше максимального уровня логического нуля (0.4 В) и меньше минимального уровня логической единицы (2.4 В). Это напряжение (1-1.5 В) подается на информационные эмиттеры, и они заперты.

При выборке данного ЗЭ на его адресные эмиттеры с выходов адресных дешифраторов поступает потенциал логической 1 ($\geq$ 2.4 В), превышающий потенциал информационных эмиттеров. Поэтому адресные эмиттеры оказываются запертыми, а коллекторный ток открытого транзистора течет через информационный эмиттер, чем обеспечивает возможность считывания и записи в него информации. Состояние ЗЭ распознается по наличию тока соответственно в разрядной линии 0 (откры-

тый Т1) или в разрядной линии 1 (открытый Т2). Считывание происходит без разрушения информации и может быть многократным.

Хранимая в ЗЭ информация доступна для считывания все время, пока ЗЭ находится в выбранном состоянии (на обеих адресных линиях выставлена логическая единица) и в него не проводится запись. Поскольку $R_{вых}$ открытого усилителя записи очень мало и шунтирует вход усилителя считывания (в него ответвляется большая часть тока считывания), в режиме считывания выходные каскады усилителя записи переводят в Z-состояние. Его $R_{вых}$ при этом резко повышается, причем $R_{вых_{зап}} >> R_{вх_{счит}}$, т.е. весь ток информационного эмиттера протекает через усилитель считывания. Таким образом, на выходах усилителей считывания появится соответствующий состоянию выбранного ЗЭ парафазный сигнал.



Зп0(1) – усилитель записи 0(1);
Чт0(1) – усилитель чтения 0(1)

Рис. 4.10. ЗЭ полупроводникового биполярного ЗУ

В режиме записи на входы усилителей записи синхронно с импульсами выборки подаются парафазные сигналы соответствующего символа (0 или 1). Так, для записи 1 в ЗЭ необходимо подать на вход левого усилителя записи (Зп0) 0, а на вход правого усилителя записи (Зп1) – 1. Для записи в ЗЭ нуля – все наоборот.

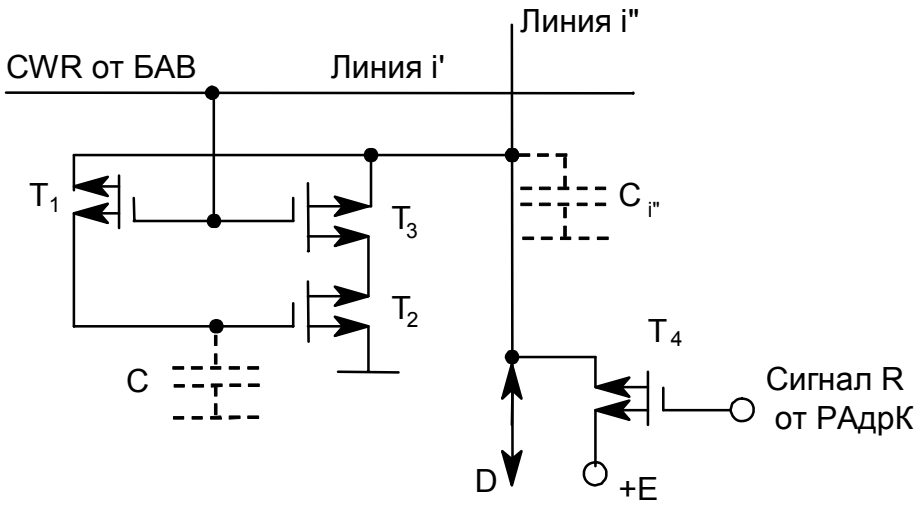
Пример 1
В ЗЭ записан 0, т.е. T_2 заперт, T_1 открыт. Происходит запись в ЗЭ единицы. С учетом инверсии в усилителе записи на эмиттер 21 попадает низкий уровень, а на эмиттер 11 – высокий. В результате T_1 закрывается, T_2 – открывается и ЗЭ переходит в состояние 1.

Пример 2
В ЗЭ записан 0, т.е. T_2 заперт, T_1 открыт. Происходит запись в ЗЭ ноля, т.е. на вход левого усилителя записи (Зп0) подается 1, а на вход правого (Зп1) – 0. В результате на эмиттер 21 попадает высокий уровень, а на эмиттер 11 – низкий. При этом состояния T_1 и T_2 не изменяются.

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

МОП-ЗЭ

Структура ЗЭ динамического МОП-ЗУ приведена на рис. 4.11. Запоминающее устройство такого типа строится по схеме 2D-M.



CWR – Control write/read;
БАВ – блок адресной выборки;
РАдрК – разрядно-адресный коммутатор

Рис. 4.11. ЗЭ полупроводникового динамического МОП-ЗУ

Как уже отмечалось, в ЗУ типа 2D-M адрес ячейки i делится на две части i′ и i″, которые соответственно поступают на БАВ и РАдрК. Запоминающей емкостью служит паразитная емкость C затвора транзистора T_2. Линия разрядно-адресного коммутатора i″ используется для ввода в ЗЭ бита информации при записи и съема его при считывании. Поскольку ЗЭ использует источник питания только при считывании, то им может служить паразитная емкость $C_{i''}$ линии i″.

Предварительно перед считыванием от РАдрК подается сигнал R, с помощью которого подготавливается считывание с мультиплексированием для ЗЭ, выбираемых линией i′. Сигнал R открывает транзистор T_4, и емкость $C_{i''}$ подзаряжается от источника +E. Затем на линию i′ подается от БАВ сигнал считывания – промежуточный уровень сигнала CWR (Control write/read), который открывает транзистор T_3, но не может открыть T_1. Пусть емкость C заряжена (т.е хранит 1) и транзистор T_2 открыт. В этом случае через открытые транзисторы T_3 и T_2 конденсатор $C_{i''}$ разряжается и низкий уровень (уровень 0) сигнала D на линии i″ указывает, что ЗЭ хранил инверсное значение, т.е. 1. Если ЗЭ хранит 0, то емкость C разрежена, T_2 закрыт и сигнал CWR не может вызвать разряд емкости $C_{i''}$. Высокий уровень сигнала D (уровень 1) указывает, что ЗЭ хранил 0. Далее сигнал D через разрядно-адресный коммутатор поступает на выход ЗУ.

При записи на линию i″ поступает сигнал D, соответствующий записываемому двоичному символу. Затем на линию i′ подается высокий уровень сигнала CWR, открывающий транзистор T_1, который подключает к линии i″ конденсатор C. В результате независимо от своего предыдущего состояния емкость оказывается заряженной, если записывается 1, и разряженной, если записывается 0.

Следует отметить, что ЗЭ динамических ЗУ имеют разную сложность и количество используемых транзисторов. В настоящее время наиболее часто используются ЗЭ, построенные на одном транзисторе.

Независимо от типа ЗЭ динамические ЗУ требуют периодической регенерации. Первоначально операциями регенерации памяти занимался процессор. Однако по мере развития элементной базы ЭВМ функции регенерации памяти стали выполняться на более низком уровне. Для регенерации стали использовать один из каналов контроллера прямого доступа к памяти (см. п. 11), а затем только контроллер памяти. В настоящее время схемы регенерации во многих случаях располагаются непосредственно в самом кристалле памяти, и от разработчика не требуется специальных мер по организации этого процесса. Такие ЗУ часто называют квазистатическими. Между тем процесс регенерации информации в отдельных БИС памяти все равно требует некоторой синхронизации. Эта задача в ЭВМ различной архитектуры решается по-разному, в частности в IBM PC контроль над процессом регенерации памяти включен в функции *чипсета* (см. п. 10).

4.5. ПОСТОЯННЫЕ ЗУ (ПЗУ, ППЗУ)

Постоянные ЗУ в рабочем режиме ЭВМ допускают только считывание хранимой информации. В зависимости от типа ПЗУ занесение в него информации производится или в процессе изготовления, или в эксплуатационных условиях путем настройки, предваряющей использование ПЗУ в вычислительном процессе. В последнем случае ПЗУ называются постоянными запоминающими устройствами с изменяемым в процессе эксплуатации содержимым или программируемыми постоянными запоминающими устройствами (ППЗУ).

Постоянные ЗУ обычно строятся как адресные. Функционирование ПЗУ можно рассматривать как выполнение однозначного преобразования k-разрядного кода адреса ячейки запоминающего массива ЗМ в n-разрядный код хранящегося в ней слова.

По сравнению с ЗУ с произвольным обращением, допускающим как считывание, так и запись информации, конструкции ПЗУ значительно проще, их быстродействие и надежность выше, а стоимость ниже. Это объясняется большей простотой ЗЭ, отсутствием цепей для записи вообще или, по крайней мере, для оперативной записи, реализацией неразрушающего считывания, исключающего процедуру регенерации информации.

Одним из важнейших применений ПЗУ является хранение микропрограмм в микропрограммных управляющих устройствах ЭВМ. Для этой цели необходимы ПЗУ значительно большего, чем в ОП, быстродействия и умеренной емкости (10 000 - 100 000 бит).

Постоянные ЗУ широко используются для хранения программ в специализированных ЭВМ, в том числе в микроЭВМ, предназначенных для решения определенного набора задач, для которых имеются отработанные алгоритмы и программы, например в бортовых ЭВМ самолетов, ракет, космических кораблей, в управляющих вычислительных комплексах, работающих в АСУ технологических процессов. Такое применение ПЗУ позволяет существенно снизить требования к емкости ОП, повысить надежность и уменьшить стоимость вычислительной установки.

Очень широко ПЗУ используются в универсальных ЭВМ всех классов для хранения стандартных процедур начальной инициализации вычислительной системы и

внешних устройств, например BIOS в PC фирмы IBM. Программное обеспечение контроллеров интеллектуальных внешних устройств ЭВМ обычно также хранится во встроенных ПЗУ.

На рис. 4.12 приведена схема простейшего ПЗУ со структурой типа 2D. Запоминающий массив образуется системой взаимно перпендикулярных линий, в пересечениях которых устанавливаются ЗЭ, которые либо связывают (состояние 1), либо не связывают (состояние 0) между собой соответствующие горизонтальную и вертикальную линии. Поэтому часто ЗЭ в ПЗУ называют связывающими элементами. Для некоторых типов ЗЭ состояние 0 означает просто отсутствие запоминающего (связывающего) элемента в данной позиции ЗМ.

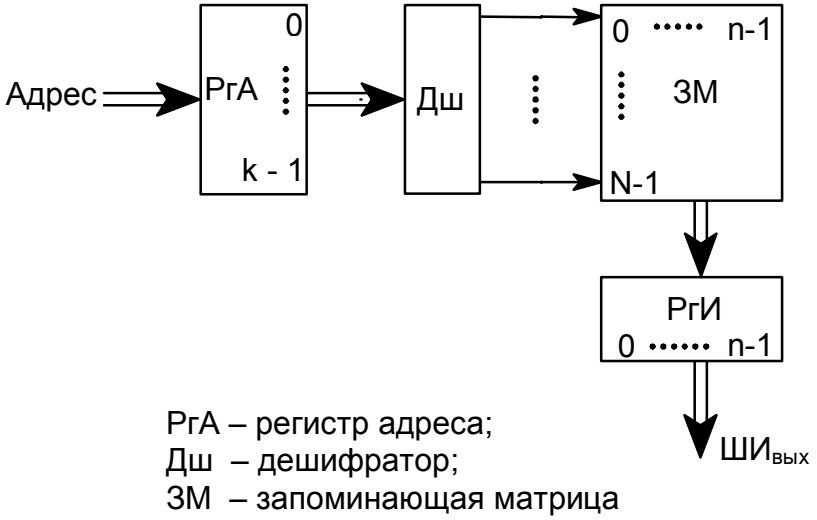


Рис. 4.12. Постоянное ЗУ типа 2D

Дешифратор ДШ по коду адреса в РгА выбирает одну из горизонтальных линий (одну из ячеек ЗМ), в которую подается сигнал выборки. Выходной сигнал (сигнал 1) появляется на тех вертикальных разрядных линиях, которые имеют связь с возбужденной адресной линией. В зависимости от типа запоминающих (связывающих) элементов различают резисторные, емкостные, индуктивные (трансформаторные), полупроводниковые (интегральные) и другие ПЗУ.

В настоящее время наиболее распространенным типом являются полупроводниковые интегральные ПЗУ.

Полупроводниковые интегральные ПЗУ имеют все те же достоинства, что и полупроводниковые ЗУ с произвольным обращением. Более того, в отличие от последних они являются энергонезависимыми. Постоянные ЗУ имеют большую емкость на одном кристалле (в одном корпусе интегральной микросхемы). Положительным свойством интегральных ПЗУ является то, что некоторые типы этих устройств позволяют самому потребителю производить их программирование (занесение информации) в условиях эксплуатации и даже многократное перепрограммирование.

По типу ЗЭ, устанавливающих или разрывающих связь (контакт) между горизонтальными и вертикальными линиями, различают биполярные и МОП-схемы ПЗУ. Биполярные ПЗУ имеют время выборки 3-5 нс. Постоянные ЗУ на МОП-схемах имеют большую емкость в одном кристалле (корпусе), но и меньшее быстродействие: время выборки 10-15 нс.

По важнейшему признаку – способу занесения информации (программированию) различают три типа интегральных полупроводниковых ПЗУ:

- Программирование в процессе изготовления путем нанесения при помощи фотошаблонов в нужных потребителю точках контактных перемычек.

- Программирование путем выжигания перемычек или пробоя p-n переходов для уничтожения или образования связей между горизонтальными и вертикальными линиями (одноразовое программирование), которое может осуществить сам пользователь с помощью специального программатора.
- Электрическое перепрограммирование, при котором информация заносится в ЗМ электрическим путем, а стирание информации, необходимое для изменения содержимого ПЗУ, выполняется воздействием на ЗМ ультрафиолетовым излучением или электрическим путем (многократное программирование). Время программирования для обоих типов ППЗУ примерно одинаково и составляет около 30-100 с на 1 мегабит памяти.

Программируемые фотошаблонами и выжиганием ПЗУ могут строиться на основе как биполярных, так и МОП-схем. Перепрограммируемые ПЗУ используют только МОП-схемы, способные хранить заряды.

Различные типы ЗЭ интегральных ПЗУ представлены на рис. 4.13. На рис. 4.13,а показан биполярный транзисторный ЗЭ с выжигаемой перемычкой, соединяющей горизонтальную и вертикальную линии. При программировании ПЗУ перемычки выжигаются в нужных местах импульсами тока с амплитудой 20-30 мА. При выборе адресным дешифратором горизонтальной линии X на базу транзистора ЗЭ поступает открывающий его сигнал, и при наличии перемычки (состояние 1) на вертикальной линии Y появится потенциал коллектора транзистора +5 В.

На рис. 4.13,б изображен ЗЭ, программируемый пробиванием p-n перехода. В исходном состоянии включенные встречно диоды изолируют линии X и Y (состояние 0). При подаче повышенного напряжения диод Д2 пробивается и закорачивается (состояние 1).

Более просто устроены ПЗУ с транзисторными и диодными запоминающими (связывающими) элементами, программируемые при изготовлении ПЗУ. В этом случае с помощью фотошаблонов в нужных позициях ЗМ наносятся или не наносятся контактные перемычки (вместо плавкой перемычки или вместо диода Д1 на рис. 4.13,а и б соответственно).

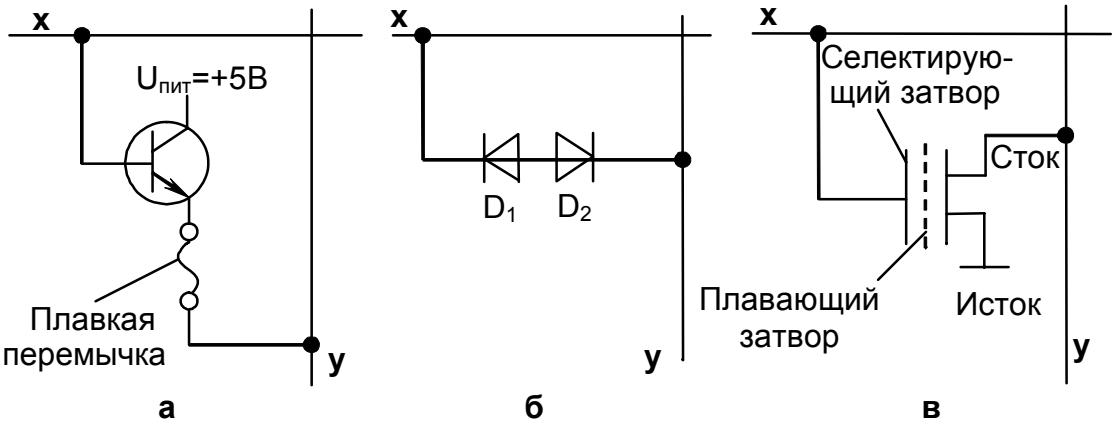


Рис. 4.13. Запоминающие (связывающие) элементы программируемых интегральных полупроводниковых постоянных ЗУ:
а – элемент с плавкой (выжигаемой) перемычкой;
б – элемент с пробиваемым p-n-переходом;
в – лавинно-инжекционный МОП-транзистор с плавающим и селектирующим затворами

На рис. 4.13,в представлен ЗЭ в виде лавинно-инжекционного МОП-транзистора с плавающим и селектирующим затворами. Интегральные ПЗУ на таких элементах допускают многократную замену хранимой информации.

Плавающий (изолированный) затвор не имеет электрического подвода, он предназначен для хранения заряда. Селектирующий затвор подсоединен к одному из выходов дешифратора строк – к горизонтальной линии, а сток – к вертикальной линии. В исходном состоянии отсутствует заряд на плавающем затворе (состояние 1), транзистор имеет очень небольшое пороговое напряжение. Выбор элемента осуществляется путем подачи на селектирующий затвор выходного напряжения адресного дешифратора, при этом включается транзистор и через цепь сток-исток протекает значительный ток. Программирование (занесение 0 в элементы) производится подачей на сток импульса напряжения 25-30В. При этом происходит инжекция электронов, имеющих высокую энергию, через оксид на изолированный затвор, получающий отрицательный заряд (состояние 0). В результате увеличивается пороговое напряжение, и подача на селектирующий затвор выходного напряжения дешифратора не включает этот транзистор. Сообщенное элементу состояние сохраняется сколь угодно долго.

4.6. ФЛЭШ-ПАМЯТЬ

Флэш-память (flash-memory) по типу запоминающих элементов и основным принципам работы подобна памяти типа EEPROM (ППЗУ) с электрическим перепрограммированием. Однако ряд архитектурных и структурных особенностей позволяют выделить ее в отдельный класс. Разработка флэш-памяти считается кульминацией развития схемотехники памяти *с* электрическим стиранием информации, и стала возможной только после создания технологий сверхтонких пленок. Время электрического перепрограммирования флэш-памяти в отличие от существующих ППЗУ очень мало и составляет сотни наносекунд. Это позволяет использовать их в качестве оперативных внешних запоминающих устройств типа жесткого диска. Однако число циклов перезаписи флэш-памяти ограничено.

В схемах флэш-памяти не предусмотрено стирание отдельных слов, стирание информации осуществляется либо для всей памяти одновременно, либо для достаточно больших блоков. Это позволяет упростить схемы ЗУ, т. е. способствует достижению высокого уровня интеграции и быстродействия при снижении стоимости. Технологически схемы флэш-памяти выполняются с высоким качеством и обладают очень хорошими параметрами.

Термин flash, по одной из версий, связан с характерной особенностью этого вида памяти – возможностью одновременного стирания всего ее объема. Согласно этой версии еще до появления флэш-памяти при хранении секретных данных использовались устройства, которые при попытках несанкционированного доступа к ним автоматически стирали хранимую информацию и назывались устройствами типа flash (вспышка, мгновение). Это название перешло и к памяти, обладавшей свойством быстрого стирания всего массива данных одним сигналом.

Одновременное стирание всей информации ЗУ реализуется наиболее просто, но имеет тот недостаток, что даже замена одного слова в ЗУ требует стирания и новой записи для всего ЗУ в целом. Для многих применений это неудобно, поэтому наряду со схемами с одновременным стиранием всего содержимого имеются схемы с блочной структурой, в которых весь массив памяти делится на блоки, стираемые независимо друг от друга. Объем таких блоков сильно разнится: от 256 байт до 128 Кбайт.

Число циклов перепрограммирования для флэш-памяти хотя и велико, но ограничено, т.е. ячейки при перезаписи "изнашиваются". Для того, чтобы увеличить долговечность памяти, в ее работе используются специальные алгоритмы, способствующие "разравниванию" числа перезаписей по всем блокам микросхемы.

Соответственно областям применения флэш-память имеет архитектурные и схемотехнические разновидности. *Двумя основными направлениями эффективного*

использования флэш-памяти являются хранение не очень часто изменяемых данных (обновляемых программ, в частности) и замена памяти на магнитных дисках.

Для первого направления, в связи с редким обновлением содержимого, параметры циклов стирания и записи не столь существенны, как информационная емкость и скорость считывания информации. Стирание в этих схемах может быть как одновременным для всей памяти, так и блочным. Среди устройств с блочным стиранием выделяют схемы со специализированными блоками – несимметричные блочные структуры – по имени так называемых boot-блоков, в которых информация надежно защищена аппаратными средствами от случайного стирания. Эти ЗУ называют *boot block flash memory.* Boot-блоки хранят программы инициализации системы, позволяющие ввести ее в рабочее состояние после включения питания.

Микросхемы для замены жестких магнитных дисков *(flash-file memory)* содержат более развитые средства перезаписи информации и имеют идентичные блоки (симметричные блочные структуры). Накопители подобного типа широко используются фирмой Intel. Имеются мнения о конкурентоспособности этих накопителей в применениях, связанных с заменой жестких магнитных дисков для ЭВМ различных типов.

В заключение следует отметить, что в настоящем разделе рассмотрены только основные типы ЗУ и ЗЭ, которые далеко не исчерпывают все разнообразие современной элементной базы устройств памяти ЭВМ.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Перечислите характеристики ЗУ.
2. Назовите основные критерии, по которым можно классифицировать устройства памяти.
3. Почему в современных ЭВМ память имеет иерархическую структуру?
4. Перечислите общие принципы построения иерархической памяти.
5. Изобразите и опишите иерархическую структуру памяти.
6. Что представляет собой ОП?
7. Для чего нужна внешняя память?
8. Опишите назначение кэш-памяти.
9. Что такое СОП?
10. Кратко охарактеризуйте каждый способ организации памяти.
11. Опишите структуру адресной памяти.
12. Каков принцип построения ассоциативной памяти?
13. Изобразите структурную схему ассоциативной памяти, объясните назначение каждого блока в этой схеме.
14. Опишите принципы построения стековой памяти.
15. Какова структура ЗМ?
16. Приведите структуру ЗУ типа 2D.
17. Опишите ЗУ типа 3D.
18. Изобразите структуру ЗУ типа 2D-М и опишите принцип его работы.
19. По какому принципу построены ЗЭ на ферритовых кольцах?
20. Опишите ЗУ с ЗЭ, построенными на биполярных транзисторах.
21. Приведите структуру ЗЭ динамического МОП-ЗУ.
22. В чем различие между ПЗУ и ППЗУ?
23. Опишите различные типы ЗЭ интегральных ПЗУ.
24. Приведите схему ПЗУ типа 2D. Опишите принцип его действия.

КОНТРОЛЬНЫЕ ЗАДАНИЯ

1. На листах ответа должны быть указаны номер группы, фамилия студента и номер его варианта.

2. Номера вопросов выбираются студентом в соответствии с двумя последними цифрами в его зачетной книжке. В табл.4.1 a_{n-1} – это предпоследняя цифра номера, a_n – последняя цифра. В клетках таблицы стоят номера вопросов, на которые необходимо дать письменный ответ.

Номера вопросов **Таблица 4.1**

a_{n-1} \ a_n	0	1	2	3	4	5	6	7	8	9
0	1,6,10, 15,19	2,7,11, 18,22	3,8,12, 16,21	4,9,13, 17,23	5,8,14, 18,19	1,7,11, 16,20	2,6,10, 16,24	3,8,12, 15,19	4,9,12, 15,23	5,7,11, 15,22
1	3,6,14, 16,22	4,9,13, 16,19	1,6,10, 18,21	2,7,11, 15,23	3,6,13, 15,21	5,8,14, 17,24	4,9,13, 15,20	1,7,13, 18,22	5,9,13, 17,24	2,8,11, 17,19
2	4,7,13, 16,24	1,6,10, 17,20	3,8,12, 18,23	5,6,14, 18,20	4,9,13, 18,24	2,8,14, 15,23	1,6,10, 16,22	2,7,11, 17,21	3,8,12, 17,22	4,9,10, 18,21
3	5,8,14, 16,20	2,7,11, 16,24	3,6,10, 17,23	1,7,10, 15,19	5,8,12, 16,23	4,9,14, 17,20	2,8,12, 18,20	1,7,12, 17,21	3,6,11, 18,24	5,8,14, 15,21
4	1,6,10, 15,19	2,7,11, 18,22	3,8,12, 16,21	4,9,13, 17,23	5,8,14, 18,19	1,7,11, 16,20	2,6,10, 16,24	3,8,12, 15,19	4,9,12, 15,23	5,7,11, 15,22
5	3,6,14, 16,22	4,9,13, 16,19	1,6,10, 18,21	2,7,11, 15,23	3,6,13, 15,21	5,8,14, 17,24	4,9,13, 15,20	1,7,13, 18,22	5,9,13, 17,24	2,8,11, 17,19
6	4,7,13, 16,24	1,6,10, 17,20	3,8,12, 18,23	5,6,14, 18,20	4,9,13, 18,24	2,8,14, 15,23	1,6,10, 16,22	2,7,11, 17,21	3,8,12, 17,22	4,9,10, 18,21
7	5,8,14, 16,20	2,7,11, 16,24	3,6,10, 17,23	1,7,10, 15,19	5,8,12, 16,23	4,9,14, 17,20	2,8,12, 18,20	1,7,12, 17,21	3,6,11, 18,24	5,8,14, 15,21
8	1,6,10, 15,19	2,7,11, 18,22	3,8,12, 16,21	4,9,13, 17,23	5,8,14, 18,19	1,7,11, 16,20	2,6,10, 16,24	3,8,12, 15,19	4,9,12, 15,23	5,7,11, 15,22
9	3,6,14, 16,22	4,9,13, 16,19	1,6,10, 18,21	2,7,11, 15,23	3,6,13, 15,21	5,8,14, 17,24	4,9,13, 15,20	1,7,13, 18,22	5,9,13, 17,24	2,8,11, 17,19

5. СТРУКТУРА И ФОРМАТЫ МАШИННЫХ КОМАНД, СПОСОБЫ АДРЕСАЦИИ

5.1. ОБЩИЕ ЗАМЕЧАНИЯ

При рассмотрении работы процессора подчеркивалось, что информация о том, какую машинную операцию надо выполнить в данный момент, над какими операндами и куда поместить результат, задается *машинной командой*. При этом любая программа неймановской машины представляет собой последовательность команд, отображающих все действия, необходимые для решения задачи по некоторому алгоритму.

Машинная команда представляет собой код, определяющий операцию вычислительной машины и данные, участвующие в операции. В общем случае команда должна содержать также в явной или неявной форме информацию об адресе, по которому помещается результат операции, и об адресе следующей команды.

Машинная операция – это действия машины по преобразованию информации, выполняемые под воздействием одной команды.

По характеру выполняемых операций различают следующие *основные* группы команд:

- Арифметические операции над ЧФЗ и ЧПЗ.
- Команды десятичной арифметики.

- Логические (поразрядные) операции.
- Передача кодов (пересылка операндов).
- Операции ввода-вывода.
- Управление порядком выполнения команд (передача управления).
- Задание режима работы машины и различные дополнительные действия.

В общем виде машинная команда имеет структуру, изображенную на рис. 5.1.

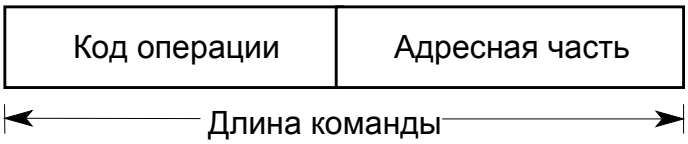


Рис. 5.1. Общая структура команды

Таким образом, команда состоит из операционной и адресной частей. Эти части, в свою очередь, могут состоять из нескольких полей (особенно адресная).

Операционная часть – содержит код, который задает вид операции (сложение, умножение, передача и т.д.).

Адресная часть – содержит информацию об адресах операндов и результата операции, а в некоторых случаях и следующей команды.

Структура команды – определяется составом, назначением и расположением полей в команде.

Формат команды – это ее структура с разметкой номеров разрядов, определяющих границы отдельных полей команды.

Задача выбора оптимальных структур и форматов команд при проектировании новых ЭВМ является одной из важнейших, поскольку от правильности ее решения зависит быстродействие и производительность ЭВМ.

Проблема состоит в том, что, с одной стороны, в команде желательно разместить максимум информации о выполняемой операции. С другой стороны, для упрощения аппаратуры и повышения быстродействия ЭВМ длина формата команды должна быть согласована с длиной обрабатываемых машинных слов, составляющей обычно 16-32 бита (для того чтобы можно было использовать для хранения и обработки операндов и команд одни и те же аппаратные средства). Формат команды должен быть, по возможности, короче, укладываться в машинное слово или полуслово, а для ЭВМ с коротким словом (8-16 бит) быть малократным машинному слову. Решение проблемы выбора оптимального формата команды значительно усложняется в микроЭВМ, работающих с коротким словом.

В абсолютном большинстве случаев ОП универсальных ЭВМ является адресной. Это значит, что каждой хранимой в ОП единице информации (байту, слову, двойному слову) ставится в соответствие специальное число – адрес, определяющий место ее хранения в памяти. В современных ЭВМ различных типов *минимальной* адресуемой в памяти единицей информации в большинстве случаев является *один байт*, т.е. 8 бит с 9-м контрольным разрядом. Иногда бывает и полубайт, т.е. 4 разряда и даже один бит. Более крупные единицы информации – слово, двойное слово и т.д. образуются из целого числа байт. В зависимости от способа хранения информации в ОП их адресом считается адрес старшего или младшего байта.

В общем случае разрядность машинного слова может определяться разрядностью АЛУ процессора, разрядностью шины данных, шириной выборки ОП и другими факторами. Наиболее часто разрядность машинного слова соответствует разрядности операндов, которые наиболее эффективно обрабатываются процессором. Например, процессор I80386 имеет 32- разрядные АЛУ и 32- разрядную шину данных. Однако разработчики устройств на базе этого процессора за машинное слово выбрали 16- разрядный двоичный код при ширине выборки ОП 1 байт. Следует иметь в виду, что ширина выборки ОП – это техническая характеристика БИСов памяти, а

байт, слово, двойное слово и т.д. – логические единицы информации, которые формируются контроллером памяти и операционной системой и обычно кратны ширине выборки. При рассмотрении процессов передачи и обработки информации внутри ЭВМ в большинстве случаев оперируют именно этими логическими единицами.

5.2. ВОЗМОЖНЫЕ СТРУКТУРЫ МАШИННЫХ КОМАНД

Процесс эволюции ЭВМ и расширение сферы их целевого использования, совершенствование аппаратного и программного обеспечения ЭВМ привели к созданию машинных команд очень сложной структуры. Однако, если отбросить детали построения команд конкретных процессоров, возможные структуры машинных команд сводятся к пяти основным типам.

1. *Четырехадресная структура*

Такая команда содержит наиболее полную информацию о выполняемой операции (рис. 5.2, а), так как она содержит поле кода операции и четыре адреса для указания ячеек памяти двух операндов, участвующих в операции, ячейки, в которую помещается результат операции, и ячейки, содержащей следующую команду. Такой порядок выборки команд называется *принудительным*. Он использовался в первых моделях ЭВМ, имеющих небольшое число команд и очень незначительный объем ОП.

Рассмотрим длину такой команды применительно к ЭВМ, имеющей порядка 200 команд и объем памяти порядка 16 Мбайт. В этом случае длина КОП будет:

$N_{КОП} = \log_2 200$, т.е. 8 разрядов ($2^8 = 256$).

Для обеспечения доступа ко всем ячейкам памяти потребуется

$$N_{адр} = \log_2 16777216 = \log_2 16 \cdot 1024 \cdot 1024 = \log_2 2^4 \cdot 2^{10} \cdot 2^{10} = \log_2 2^{24} = 24 .$$

Таким образом, длина четырехадресной команды составит

$$N_{ком} = 8 + 24 * 4 = 104 \text{ } разряда .$$

Полученная длина команды оказывается недопустимо большой, поэтому в современных ЭВМ такая структура команд не используется.

2. *Трехадресная структура*

Можно построить ЭВМ так, что после выполнения команды по адресу K (команда занимает L ячеек памяти) выполняется команда по адресу K+L. Такой порядок выборки команд называется *естественным*. Он нарушается только специальными командами передачи управления. При естественном порядке выборки адрес следующей команды формируется в устройстве, называемом *счетчик адреса команд*. В этом случае команда становится трехадресной (рис. 5.2, б).

3. *Двухадресная структура*

В большинстве случаев непосредственно перед выполнением операции операнды помещаются во внутренние регистры процессора. Можно построить аппаратную часть процессора таким образом, что результат операции будет всегда помещаться в фиксированный регистр процессора, например на место первого операнда. В этом случае команда будет двухадресной, поскольку адрес результата подразумевается (рис. 5.2, в).

4. *Одноадресная структура*

В одноадресной команде (рис. 5.2, г) подразумеваемые адреса имеют уже и результат операции, и один из операндов. Для этого аппаратная часть процессора должна быть построена так, чтобы один из операндов, например первый, и результат операции размещались в одном и том же фиксированном регистре. Выделенный для этой цели внутренний регистр процессора получил название *аккумулятор*. Адрес же другого операнда указывается в команде.

5. *Безадресная структура*

Использование безадресных команд (рис. 5.2, д) возможно только при естественном порядке выборки команд и подразумеваемых адресах обоих операндов и результата операции, например при работе со стековой памятью. В этом случае один операнд находится в вершине стека, а второй операнд – в аккумуляторе. Туда же помещается и результат.

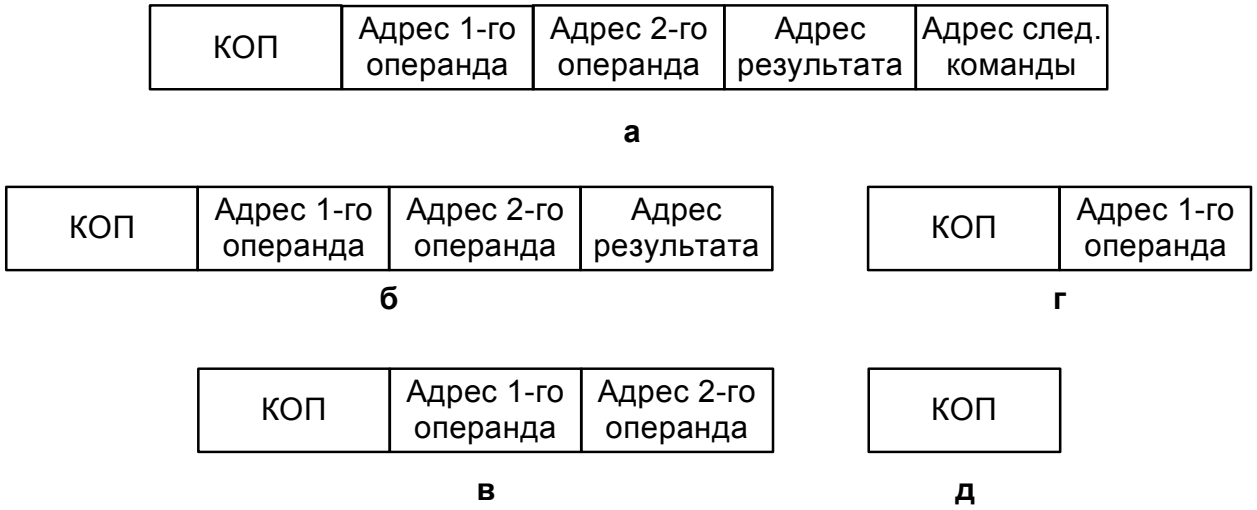


Рис. 5.2. Структуры команд: а – четырехадресная; б – трехадресная; в – двухадресная; г – одноадресная; д – безадресная

Обычно в ЭВМ используется несколько форматов команд разной длины. Кроме того, трехадресные команды в современных универсальных ЭВМ практически не используются, так как являются слишком громоздкими (хотя и самыми удобными с точки зрения программиста). Обычно используются одно- и двухадресные команды и их модификации. Следует отметить, что трехадресные команды используются при работе с памятями небольшого объема. Обычно это массивы внутренней памяти процессора. При выполнении операций типа "регистр-регистр" и достаточно больших объемах регистровой памяти (десятки регистров), как, например, в процессорах классической RISC-архитектуры (см. пп. 9.2), подобные команды очень эффективны.

Рассмотренные структуры команд достаточно схематичны. В реальных ЭВМ адресные поля команд большей частью содержат не сами адреса, а только информацию, позволяющую определить действительные (*исполнительные*) адреса операндов в соответствии с используемыми в командах способами адресации.

5.3. СПОСОБЫ АДРЕСАЦИИ

Определимся с терминами, которые будут использоваться ниже.

Адресный код (АК) – это информация об адресе операнда, содержащаяся в команде.

Исполнительный адрес (АИ) – это номер ячейки ОП, к которой производится фактическое обращение.

В большинстве команд современных ЭВМ адресный код не совпадает с исполнительным адресом. Проблема выбора способов адресации и формирования исполнительного адреса, как и системы команд, относится также к важнейшим при разработке новых процессоров и ЭВМ.

Как уже отмечалось в п. 5.2, процесс эволюции ЭВМ привел к появлению команд очень сложной структуры. Это в полной мере относится и к системам адресации, используемым в современных универсальных ЭВМ, поддерживающих многоза-

дачные режимы и виртуальную память. Между тем эти системы адресации являются сложными комбинациями относительно небольшого количества способов адресации, разработанных еще для ЭВМ первых поколений. Ниже рассмотрены основные из этих способов адресации.

Подразумеваемый операнд

В команде не содержится явных указаний о самом операнде или его адресе. Операнд подразумевается и фактически задается кодом операции команды. Этот метод используется редко, но иногда бывает очень удобен, например, в командах подсчета, когда к содержимому счетчика необходимо добавить фиксированное приращение. Это касается и других операций, связанных с добавлением (вычитанием) константы.

Подразумеваемый адрес

В команде не содержится явных указаний об адресе участвующего в операции операнда или адресе результата операции. Так, например, при выполнении данной операции результат всегда засылается по адресу второго операнда или в другой, заранее определенный, регистр. В простейших микропроцессорах результат всегда помещается в аккумулятор.

Непосредственная адресация

В команде содержится не адрес операнда, а непосредственно сам операнд (рис. 5.3). Это способ уменьшения объема программы и занимаемой памяти, так как не требует операций обращения к памяти и самой ячейки памяти.

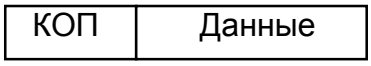
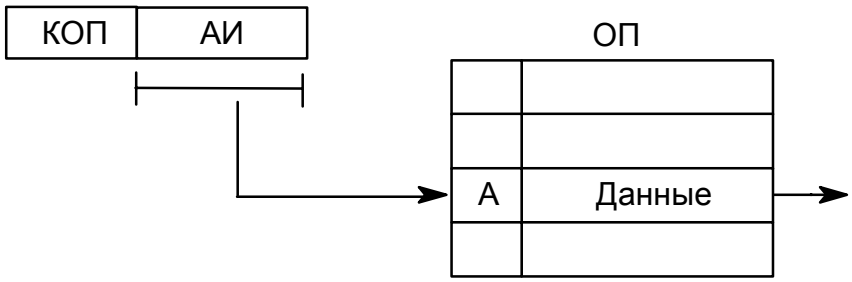


Рис. 5.3. Структура команды при непосредственной адресации

Применение данного способа ускоряет вычисления. Обычно он используется для хранения различных констант.

Прямая адресация



КОП – код операции;
АИ – исполнительный адрес;
ОП – оперативная память

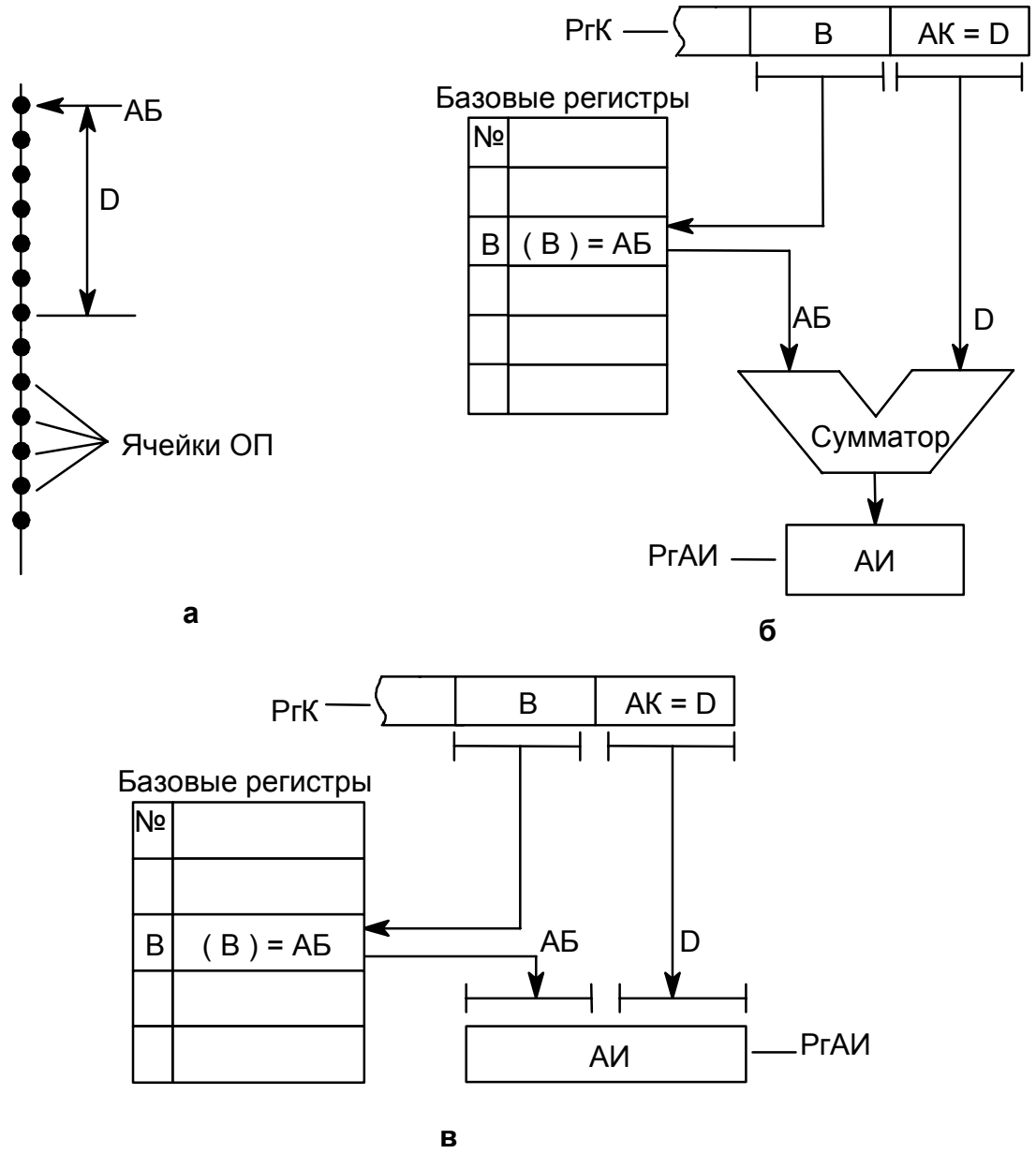
Рис. 5.4. Прямая адресация

Исполнительный адрес совпадает с адресной частью команды, т.е. адресный код совпадает с исполнительным адресом (рис. 5.4). Этот способ был основным в первых ЭВМ. В настоящее время используется в комбинации с другими способами.

Достоинства: быстрота исполнения и простота реализации. Недостатком является длинная адресная часть команды.

Относительная адресация, или базирование

Исполнительный адрес (АИ) определяется как сумма адресного кода команды (АК) и некоторого числа АБ, называемого *базовым адресом*. АИ = АБ + АК (рис. 5.5).



В	– номер базового регистра;	АИ	– исполнительный адрес;
D	– числовое значение смещения;	ОП	– оперативная память;
РгАИ	– регистр исполнительного адреса;	АБ	– базовый адрес;
ГрК	– регистр команд;	АК	– адресный код

Рис. 5.5. Базирование (относительная адресация): а – образование адреса элемента одномерного массива; б – формирование исполнительного адреса суммированием; в – формирование исполнительного адреса совмещением

Для хранения АБ используются *базовые регистры*. Это или специальные внутренние регистры процессора и сверхоперативной памяти, или специально выделенные ячейки ОП с короткими начальными адресами. В команде выделяется поле "В" для указания номера базового регистра. Число разрядов в базовом адресе АБ выбирается таким, чтобы можно было адресовать любую ячейку ОП. Адресный код АК

самой команды имеет мало разрядов и используется для представления лишь сравнительно короткого "смещения" (D). Это смещение определяет положение операнда относительно начала массива, задаваемого базовым адресом АБ (рис. 5.5, а).

Возможны два варианта формирования АИ при базировании.

- *Метод суммирования* (рис. 5.5, б).

Этот метод прост и позволяет задавать в качестве АБ любой адрес ОП. Недостатком является то, что на операцию суммирования уходит время.

- *Метод совмещения* (рис.5.5, в).

В этом случае АБ содержит старшие, а адресный код АК младшие разряды исполнительного адреса АИ, которые в регистре адреса ОП объединяются (операция *конкатенации*). При таком методе формирования АИ базовый адрес АБ может задавать не любую ячейку, а только ту, адрес которой содержит нули в младших разрядах, соответствующих смещению. Формирование адреса осуществляется быстрее, так как не требуется операции суммирования.

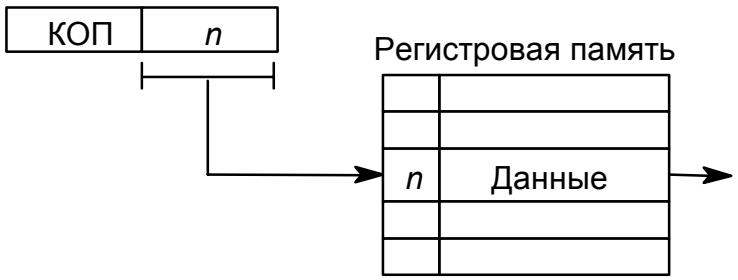
Относительная адресация обеспечивает возможность передвижения программ в памяти без изменений внутри самой программы за счет изменения базовых адресов (так называемая *перемещаемость* программ), что является основой для построения механизма виртуальной памяти (см. п. 9.4). Кроме того, она облегчает компоновку программ, части которых написаны разными программистами.

Регистровая адресация

Это частный случай так называемой *укороченной* адресации, суть которой сводится к тому, что используется только небольшая группа фиксированных ячеек памяти с начальными (короткими) адресами (0000001, 0000010, 0000011 и т.д.). Такая адресация используется только совместно с другими типами адресации.

При использовании укороченной адресации длина команды существенно сокращается, так как используются только младшие разряды адресов.

В случае регистровой адресации (рис. 5.6) в качестве фиксированных ячеек с короткими адресами используются регистры внутренней памяти процессора, которых обычно немного. Поэтому разрядность АК также невелика.



КОП – код операции

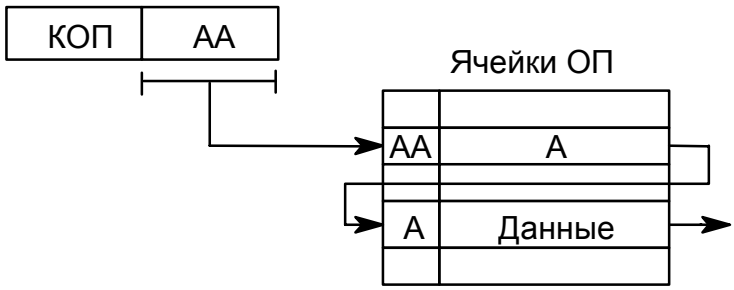
Рис. 5.6. Регистровая адресация

Это, фактически, прямая адресация к сверхбыстрой памяти процессора. Достоинства данного способа адресации – укорочение команд, увеличение скорости выполнения операций. Недостаток – малое число адресов.

Косвенная адресация

Адресный код (АК) команды указывает адрес ячейки ОП, в которой находится исполнительный адрес (АИ) операнда или команды, т.е. это адрес адреса – АА. Схема косвенной адресации представлена на рис. 5.7.

На косвенную адресацию указывает код операции (КОП) команды. В некоторых ЭВМ в команде отводится специальный разряд (*указатель адресации* – УА), и цифра 0 или 1 в нем указывает, является адресная часть команды прямым адресом или косвенным.



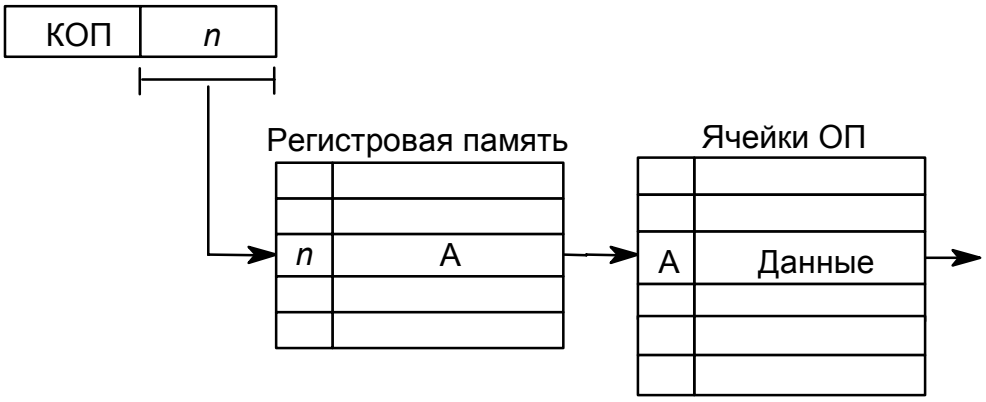
КОП – код операции;
ОП – оперативная память

Рис. 5.7. Косвенная адресация

В ряде случаев используется многоступенчатая косвенная адресация. В этом случае ячейки ОП также содержат разряд УА. Перебор ячеек ОП происходит до тех пор, пока не будет найдена ячейка, в которой УА определит прямую адресацию.

Косвенная адресация широко используется в микропроцессорах и малых ЭВМ, имеющих короткое машинное слово, для преодоления ограничений короткого формата команды. Такой способ адресации удобен также, когда требуется модификация исполнительных адресов, например, при циклической обработке элементов массива. Для этого в микропроцессорах и малых ЭВМ совместно используют *регистровую* и *косвенную* адресации (рис. 5.8).

Такая адресация позволяет микропроцессору адресоваться к ОП достаточно большого объема при небольшой длине адресного поля команды (адресного поля достаточно только для указания номеров нескольких внутренних регистров микропроцессора). Естественно, что перед выполнением этой команды в соответствующий внутренний регистр микропроцессора должен быть загружен полный адрес интересующей ячейки ОП.



КОП – код операции;
ОП – оперативная память

Рис. 5.8. Регистровая и косвенная адресации

Автоинкрементная и автодекрементная адресации

Выше отмечалось, что при косвенной регистровой адресации в соответствующий регистр перед использованием необходимо загрузить исполняемый адрес, по которому произойдет обращение к ячейке ОП. На это уходит время. Однако при обработке массива данных этих потерь можно избежать, добавив механизм автоматического приращения или уменьшения содержимого регистра при каждом обращении

к нему. В этом случае загрузка начального адреса массива происходит один раз, а затем при каждом обращении формируется адрес следующего элемента массива.

Автоинкрементная адресация – сначала (при каждом обращении) содержимое регистра используется как адрес операнда, а затем получает приращение, равное числу байт в элементе массива, т.е. формируется адрес следующего элемента.

Автодекрементная адресация – сначала содержимое соответствующего регистра уменьшается на число, равное числу байт в элементе массива, а затем используется как адрес операнда.

Следует иметь в виду, что автоинкрементная и автодекрементная адресации являются упрощенным вариантом *индексации* – весьма важного механизма преобразования исполнительных адресов команд и организации вычислительных циклов (см. п. 5.5). Эти типы адресации называют также *автоиндексацией*.

Стековая адресация

При рассмотрении устройств памяти отмечалось, что основной принцип работы стекового ЗУ соответствует правилу: "последний пришел – первый ушел" (имеется в виду стек LIFO). Это правило реализуется автоматически. Поэтому при операциях со стеком возможно безадресное задание операнда – команда не содержит адреса ячейки стека, а содержит только адрес (или он подразумевается) регистра или ячейки ОП, откуда слово загружается в стек или куда выгружается из стека. Стек может быть реализован как аппаратным путем, так и программно. В первом случае стек представляет собой одномерный массив регистров, связанных между собой разрядными цепями передачи данных. Обычно он снабжен счетчиком стека, по содержимому которого можно контролировать переполнение стека. Во втором случае стек организуется на последовательно расположенных ячейках ОП. Для его реализации требуется еще один регистр – указатель стека, в котором хранится адрес вершины, т.е. последней занятой ячейки ОП из массива ячеек, отведенных под стек.

Стек является эффективным элементом архитектуры современных ЭВМ, позволяющим во многих случаях существенно повысить скорость обработки информации. В универсальных ЭВМ общего применения (таких как персональный компьютер) программисту в большинстве случаев доступен только программный стек. На рис. 5.9 приведена схема записи числа в "перевернутый" программный стек, который используется наиболее широко.

При выполнении команды загрузки слова в стек (содержимое РгК) из регистра (в данном случае из Рг5) или ячейки ОП сначала содержимое указателя стека (УС) уменьшается на 1 (стек "перевернутый"), а затем слово помещается в ячейку стека, указываемую УС. При выгрузке слова из стека в регистр или ОП слово сначала извлекается из вершины стека, а затем УС увеличивается на 1 (на рисунке не показано).

При надлежащем расположении операндов в стеке можно произвести вычисления полностью безадресными командами. Команды этого типа обычно инициируют извлечение из стека одного или двух операндов (слов), выполнение над ними указанных в коде команды операций и запись результата в какой-либо фиксированный регистр, например, аккумулятор. Вычисления с использованием стековой памяти удобно описывать и программировать с помощью инверсной (бесскобочной) записи арифметических выражений.

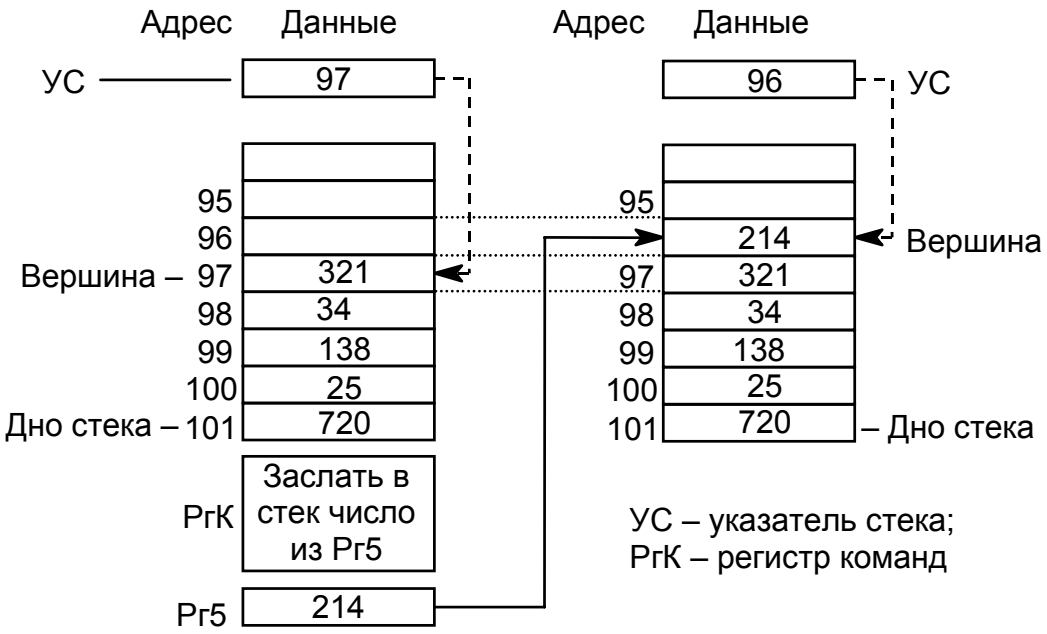


Рис. 5.9. Операция записи числа в "перевернутый" программный стек

Безадресные команды на основе стековой адресации предельно сокращают формат команд, экономят память и повышают производительность ЭВМ.

В современных ЭВМ (микропроцессорах) стек и стековая адресация широко используется для:

- сохранения содержимого регистров при переходе к подпрограмме и выходе из нее;
- сохранения информации, содержащейся во внутренних регистрах процессора при прерываниях программы;
- организации хранения элементов массивов при их циклической обработке.

Как уже отмечалось, современные ЭВМ во многих случаях используют сложные, комбинированные системы адресации, которые не могут быть в чистом виде отнесены к какому-либо одному из рассмотренных выше способов адресации. Это позволяет программисту более гибко и эффективно использовать все ресурсы ЭВМ. Однако разнообразие форматов команд и их длины ведет к усложнению УУ процессора и замедлению процесса выполнения команды.

5.4. КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

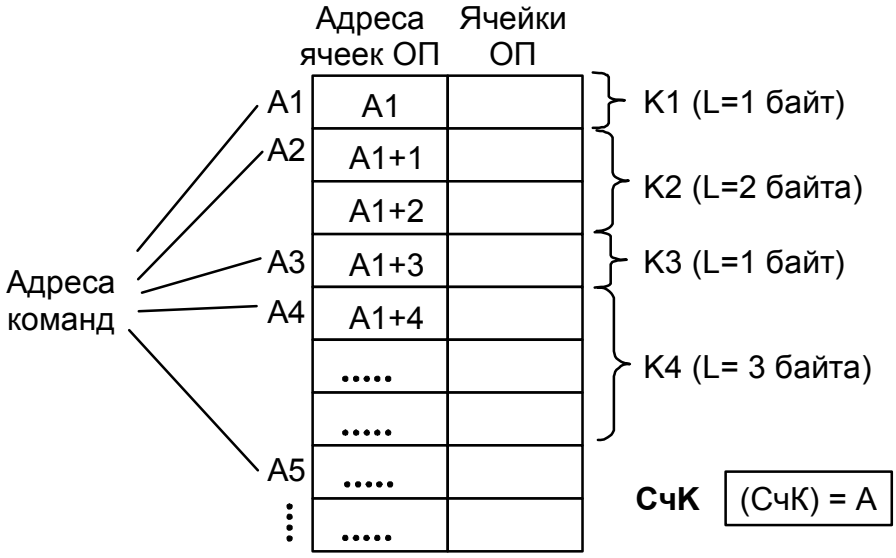
Ранее уже отмечалось, что порядок выполнения команд может быть естественным и принудительным. При естественном порядке после выполнения очередной команды выбирается команда, расположенная в следующей по порядку ячейке памяти. Обычно адрес команды хранится в специальном регистре, называемом *счетчиком адреса команд* или просто *счетчиком команд* (СчК), содержимое которого после выполнения каждой команды увеличивается на 1. Если же память имеет побайтную адресацию, то на столько байт, сколько их содержит текущая команда. Цикл выборки/выполнения команд можно пояснить схемой, приведенной на рис. 5.10. Пусть L – длина команды в байтах, а память имеет побайтную адресацию. Порядок выполнения команд на рис. 5.10 следующий:

- Выборка команды по адресу A=A1 (К1).
- Дешифровка команды, в том числе определение ее длины L (L=1).
- Вычисление адреса следующей команды (СчК) = (СчК) + 1 (К2).
- Выполнение команды (К1).
- Выборка команды по адресу A2 (К2).

- Дешифровка команды, в том числе определение ее длины L (L=2).
- Вычисление адреса следующей команды (СчК) = (СчК) + 2.
- Выполнение команды (К2).

Далее циклы выборки/выполнения команд К3, К4, … повторяются.

Естественный порядок выполнения команд может быть нарушен командами *передачи управления* (командами перехода). Следует иметь в виду, что нарушение порядка выполнения команд возможно и в ряде других случаев, важнейший из которых – обработка запросов прерывания – будет рассмотрен в дальнейшем (см. п. 6).



L – длина команды в байтах (L=1,2,3);
СчК – счетчик адреса команд;
(СчК) – содержимое счетчика адреса команд

Рис. 5.10. Порядок выполнения команд

Известны многочисленные варианты команд перехода, однако общий принцип состоит в том, что адресная часть команды перехода непосредственно или после суммирования с содержимым базового регистра загружается в СчК. В результате после выполнения такой команды может быть выполнена команда из любой ячейки памяти, определяемой адресной частью команды перехода. Для упрощения рассмотрим команды перехода без относительной адресации. Кроме того, предполагается, что память имеет побайтную адресацию.

5.4.1. КОМАНДЫ БЕЗУСЛОВНОГО ПЕРЕХОДА (БП)

Общая структура команды безусловного перехода изображена на рис. 5.11. При исполнении этой команды переход осуществляется всегда независимо от каких-либо условий.

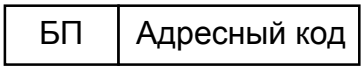
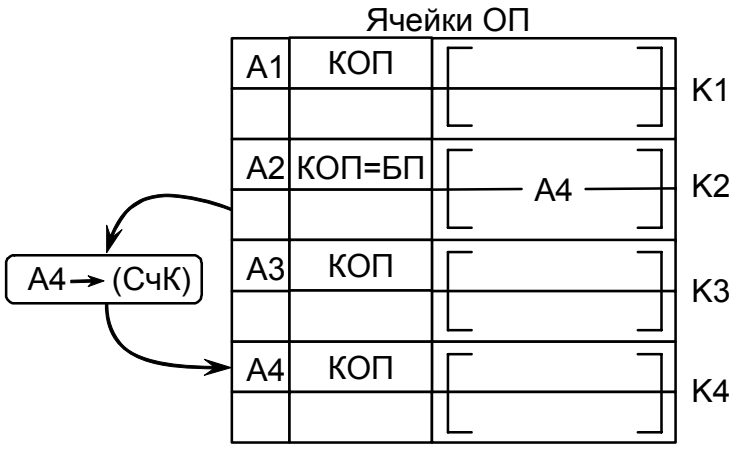


Рис. 5.11. Общая структура команды БП

Рассмотрим два возможных варианта реализации команд БП – переход по прямому и косвенному адресам.

Переход по прямому адресу

В данном примере и далее рассматриваются команды длиной 2 байта (L = 2). При выполнении команды К2 в счетчик команд загружается адрес А4, т.е. (СчК) = А4. После этого процессор начинает выполнять команды с адреса А4. Таким образом, последовательность выполнения команд следующая: К1 → К2 → К4 и далее по порядку.



ОП – оперативная память;
БП – безусловный переход;
КОП – код операции;
СчК – счетчик адреса команд;
(СчК) – содержимое счетчика адреса команд;
[ ] – адресная часть команды

Рис. 5.12. Переход по прямому адресу

Переход по косвенному адресу

Общая структура команды изображена на рис. 5.13. На косвенную адресацию указывает код операции команды БПК (рис. 5.13, а) или специальное поле К в структуре команды (рис. 5.13, б), определяющее тип адресации.

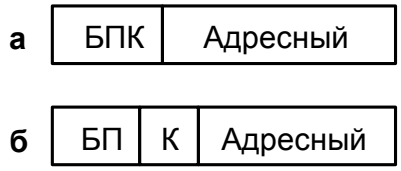
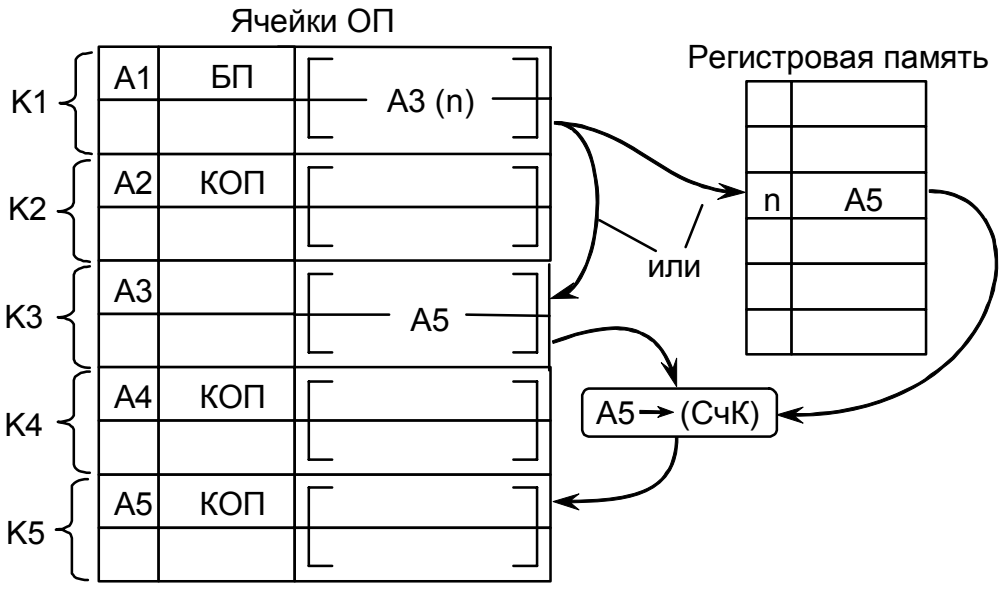


Рис. 5.13. Общая структура команды БПК

Управление передается команде с исполнительным адресом, хранящимся в ячейке (регистре) памяти, на адрес которой (которого) указывает адресное поле команды БПК (рис. 5.14). Преимущества косвенной регистровой адресации были описаны ранее. Следует иметь в виду, что К3 в ОП не является собственно командой по адресу А3 (рис. 5.14). По адресу А3 может не быть никакого КОП, поскольку здесь хранится только исполнительный адрес (АИ).

Как уже отмечалось, в современных ЭВМ широко используется относительная адресация при выполнении команд переходов всех типов, т.е. АИ формируется с учетом содержимого базового регистра.



ОП – оперативная память;
КОП – код операции;
(СчК) – значение счетчика команд;
БП – безусловный переход

Рис. 5.14. Переход по косвенному адресу

5.4.2. КОМАНДЫ УСЛОВНОГО ПЕРЕХОДА (УП)

В этом случае адрес следующей команды зависит от выполнения некоторого условия. Обычно если условие выполняется, то происходит передача управления. Если условие не выполняется, то берется следующая по порядку команда, адрес которой определяется содержимым СчК, увеличенным на приращение адреса команды (L). Это можно записать следующим образом:

$$(\text{СчК}) = \begin{cases} f\ (\text{Адр. код}) – \text{усл. выполняется} \\ (\text{СчК}) + L – \text{усл. не выполняется (естеств. порядок)} \end{cases}$$

Как и в командах БП, в командах УП используется прямая, косвенная и относительная адресации. Вид функции f зависит от используемого способа адресации. Именно команды УП позволяют строить ветвящиеся и циклические программы.

Условия перехода задаются либо самим кодом операции УП, либо в виде отдельного поля команды – *маски условия* (кода условия). В последнем случае общий формат команды УП имеет вид, показанный на рис. 5.15.

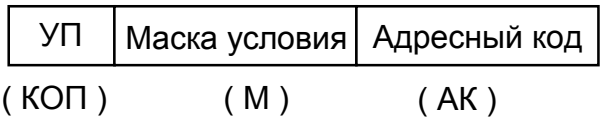
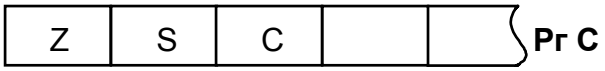


Рис. 5.15. Структура команды условного перехода

В качестве условия перехода в большинстве случаев используется тот или иной признак результата операции, выполненной под воздействием предыдущей команды. В простейших процессорах аккумуляторного типа признаки выполненной операции заносятся в разряды специального регистра процессора, называемого *регистром состояния* РгС (регистром признаков, регистром флажков). Туда же зано-

сят признаки результата выполнения некоторых операций в РОНах. Аналогичные регистры используются и в более сложных процессорах. На рис. 5.16, в качестве примера, изображена часть разрядов РгС простейшего процессора КР580ВМ80 (аналог I8080).



Z = 0 – результат ≠ 0
Z = 1 – результат = 0
} Признак 0-го результата

S = 0 – результат > 0
S = 1 – результат < 0
} Признак знака

C = 0 – нет переноса
C = 1 – есть перенос
} Признак переноса

Рис. 5.16. Регистр состояния

Команды УП определяют тип признака, по которому требуется осуществить переход (передачу управления). На тип признака может указывать сам КОП условного перехода либо 1 в соответствующем разряде маски условия (рис. 5.17).

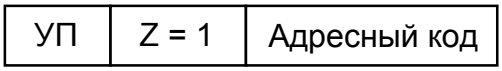


Рис. 5.17. Возможная структура команды УП

В данной команде УП маска указывает на признак нулевого результата. Это означает, что переход по программе (передача управления) осуществится при нулевом результате операции в аккумуляторе или РОНе. В общем случае маска условия может указывать на несколько признаков (например, 4- разрядная маска). В этом случае условием перехода будет дизъюнкция признаков.

Схемы выполнения команды УП при использовании прямой и косвенной адресации очень похожи на схемы, приведенные на рис. 5.12 и 5.14 для команды БП. Необходимо добавить только операцию проверки выполнения условия перехода.

Студенту рекомендуется нарисовать их самостоятельно.

5.4.3. КОМАНДЫ ПЕРЕХОДА НА ПОДПРОГРАММУ

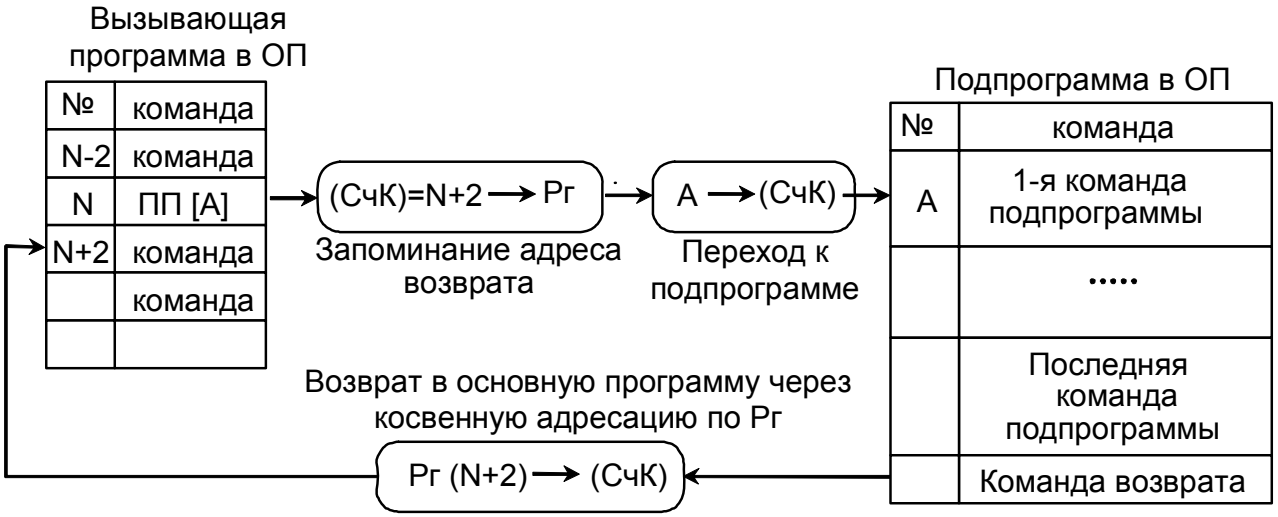
Подпрограмма представляет собой фрагмент программы, обращение к которому может иметь место в любой точке главной программы. Для перехода к подпрограмме в ЭВМ существуют команды *безусловного* и *условного переходов к подпрограмме* (ПП и ППУ). Их особенностью является то, что помимо перехода они должны обеспечить по окончании подпрограммы возврат к исходной программе, к той ее точке, откуда был совершен переход.

Ниже будут рассмотрены *только* команды *безусловного* перехода к подпрограмме (ПП), поскольку на практике они встречаются наиболее часто. Кроме того, отличие команд ПП и ППУ такое же, как и отличие команд БП и УП, т.е. перед выполнением команды ППУ происходит проверка какого-либо признака результата из РгС (регистр состояния).

Рассмотрим подробнее операции, необходимые для выполнения команды ПП в предположении, что длина команды L = 2 байта и используется прямая адресация. Эти операции поясняются схемой, приведенной на рис. 5.18.

Перед выполнением команды ПП формируется адрес возврата ($A_{возвр}$), т.е. (СчК) = (СчК) + L (в данном случае это адрес N+2). Затем $A_{возвр}$ запоминается в ячейке памяти (регистре), адрес которой (которого) в явной или неявной форме указан в команде ПП. Затем в СчК заносится содержимое адресного поля команды ПП, т.е.

адрес начала подпрограммы (А). В конце подпрограммы размещается команда возврата, которая фактически является командой БПК (безусловный косвенный переход), и указывает путем косвенной адресации (через ячейку ОП или регистр) адрес ячейки ОП, куда надо возвратиться для дальнейшего выполнения основной программы. В данном случае косвенная адресация осуществляется через регистр Рг.



(СчК) – содержимое счетчика команд

Рис. 5.18. Обращение к подпрограмме

Это простейший случай обращения к подпрограмме. Однако все современные ЭВМ допускают *вложенность* подпрограмм, т.е. обращение из одной подпрограммы к другой. При этом глубина вложения может быть достаточно велика. Для организации таких вложенных подпрограмм используется *стек*. В большинстве микропроцессоров сам стек располагается в ячейках ОП, а указатель стека оформлен в виде специального регистра.

При переходе из основной программы в подпрограмму, а затем и в следующую подпрограмму, т.е. с увеличением глубины вложения, в стек каждый раз загружается текущее состояние СчК, т.е. очередной адрес возврата. Загрузка $А_{возвр}$ в стек при выполнении команды ПП во всех процессорах, включая простейшие, происходит автоматически. Затем (после выполнения каждой подпрограммы) процессор по очереди извлекает из стека адреса возврата ($А_{возвр}$), попадая в конечном итоге в необходимую точку основной программы. Очевидно, что глубина вложения в этом случае определяется числом ячеек ОП, используемых под стек. Это число ячеек всегда ограничено, поскольку эти ячейки не могут одновременно использоваться другими частями программы.

Кроме того, с командами ПП связаны также некоторые проблемы в использовании внутренних регистров процессора. Необходимо обеспечить, чтобы подпрограмма ни в одном регистре не изменяла содержимого (особенно РгС), которое после возврата потребуется вызывающей программе. Для этого содержимое регистров, используемых подпрограммой, требуется временно запомнить, а по окончании подпрограммы – восстановить. Обычно для такого временного хранения удобно также использовать стек. Служебные действия по запоминанию и восстановлению содержимого регистров можно включить в вызывающую программу или в подпрограмму. Практика программирования показала, что их удобнее включать в подпрограмму, чтобы эти операции были записаны один раз в теле подпрограммы, а не каждый раз при ее вызове. Следует отметить, что в некоторых современных процессорах при выполнении команды ПП предусмотрено автоматическое сохранение в стеке содер-

жимого не только СчК, но и определенных групп внутренних регистров, что упрощает процедуру разработки подпрограмм.

Помимо рассмотренных операций для работы подпрограммы необходимо каким-либо образом сообщить ей о размещении обрабатываемых данных (входные параметры) и формируемых результатах (выходные параметры). Это можно сделать различными способами, но в настоящем курсе этот вопрос не рассматривается.

5.5. ИНДЕКСАЦИЯ

Характерным моментом в процессе переработки информации в ЭВМ является цикличность вычислительных процессов, при которых одни и те же операции могут выполняться над различными операндами, расположенными упорядоченно в памяти (т.е. над элементами некоторых информационных массивов). На рис. 5.19 в качестве простого примера показан одномерный массив данных, в котором подлежащие обработке операнды α_0, α_1 … α_{k-1} расположены последовательно, со сдвигом на q ячеек. Для обработки элементов такого массива необходимо, чтобы исполнительные адреса команд, вызывающих эти элементы, изменялись при каждой итерации цикла. Как уже отмечалось, для этой цели можно использовать механизм косвенной адресации, что, однако, требует относительно больших временных затрат.

Для предотвращения подобных потерь времени в современных ЭВМ используется механизм автоматического изменения исполнительных адресов соответствующих команд согласно расположению в ОП обрабатываемых операндов. Такой процесс называется *модификацией* исполнительных адресов команд (в некоторых источниках – модификацией адресных частей команд). Кроме того, этот механизм позволяет на аппаратном уровне реализовать команды управления вычислительным циклом, т.е. обеспечить необходимое количество итераций цикла с последующим выходом из него. Упрощенный алгоритм функционирования этого механизма поясняется схемой на рис. 5.19.

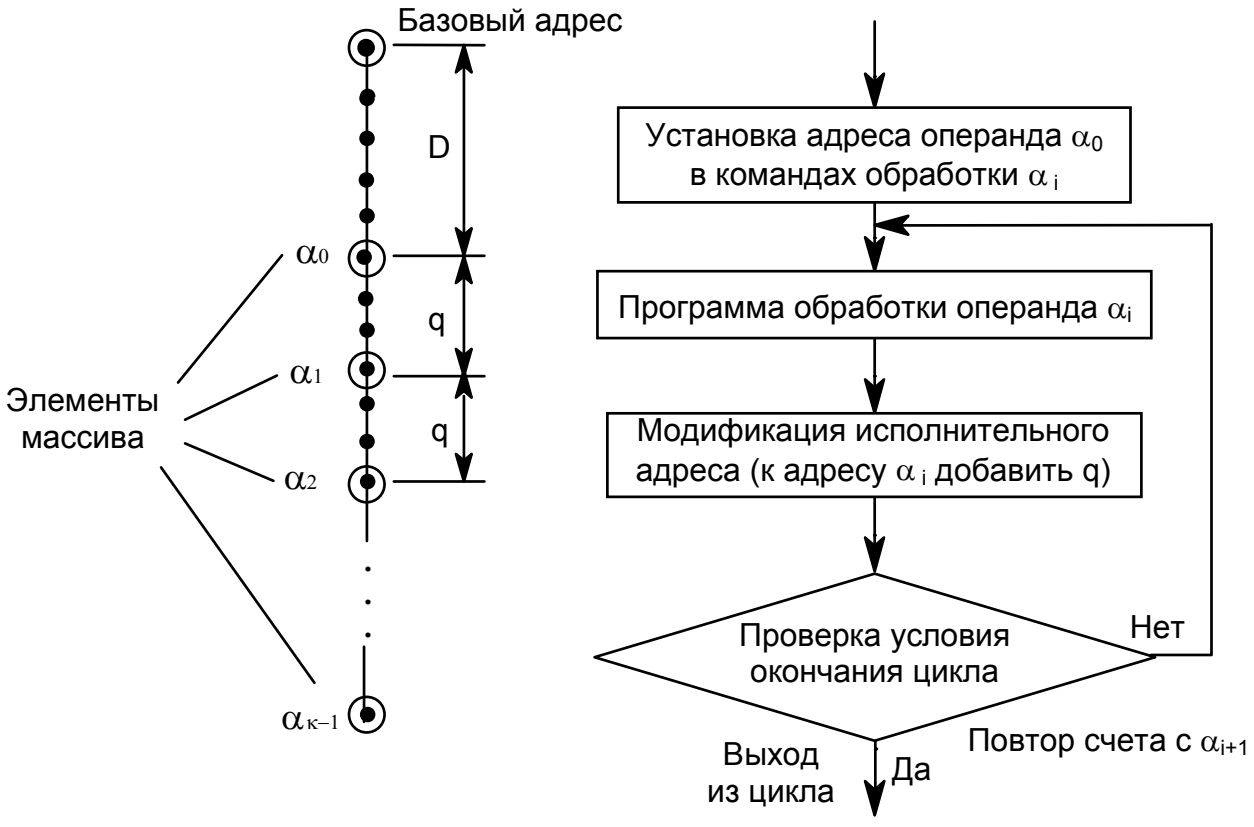


Рис. 5.19. Структура вычислительного цикла

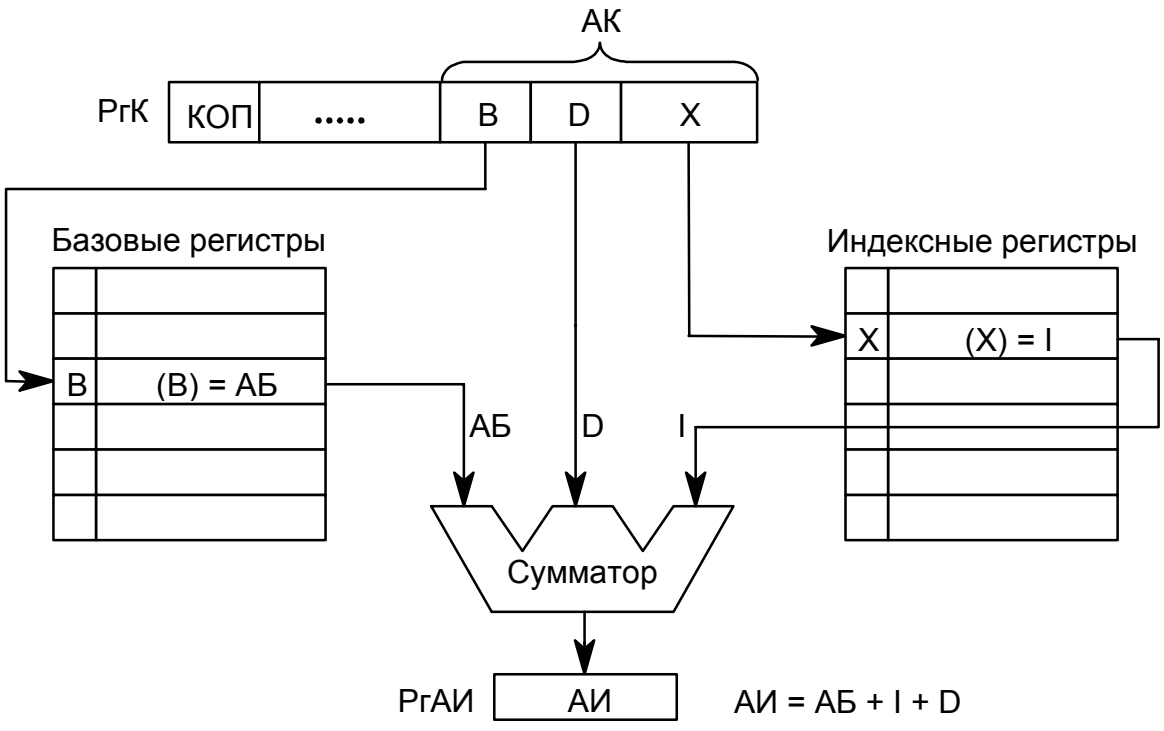
Контроль окончания циклических вычислений производится двумя способами:

- По числу итераций цикла, т.е. по содержимому некоторого счетчика, изменяемому на 1 при каждой новой итерации цикла.
- Путем задания некоторого предела для изменения модифицируемого исполнительного адреса.

Автоматическая модификация команд и управление вычислительными циклами в современных ЭВМ обеспечиваются механизмом *индексации*. Это понятие включает в себя специальный способ кодирования команд, командные и аппаратные средства задания и выполнения модификации исполнительного адреса и управления вычислительными циклами. Перечисленные средства в целом называются также *индексной арифметикой*.

Как способ модификации исполнительного адреса команды индексация является развитием механизма относительной адресации (базирования), с которым она часто используется совместно.

Для выполнения индексации в ЭВМ вводят *индексные регистры*, в качестве которых могут использоваться либо регистры внутренней памяти процессора, либо внешние по отношению к нему специальные быстродействующие регистры. В некоторых ЭВМ предусмотрена возможность, при определенных условиях, в качестве индексных регистров использовать ячейки ОП. В общем случае при использовании механизма индексации в АК команды выделяют три поля: B – номер базового регистра, D – смещение, X – номер индексного регистра.



B – номер базового регистра; АБ – базовый адрес;
D – числовое значение смещения; КОП – код операции;
АИ – исполнительный адрес; АК – адресный код;
D – числовое значение смещения; РгК – регистр команд

Рис. 5.20. Формирование АИ при индексации и базировании

Исполнительный адрес при индексации формируется путем сложения смещения (D), содержимого индексного регистра (I), а при наличии базирования – и базового адреса (АБ). Именно этот характерный

случай показан на рис. 5.20.

В ряде случаев базовые и индексные регистры назначаются программистом из некоторого общего массива регистровой памяти. Операция получения АИ выполняется либо в АЛУ процессора, либо в специальном сумматоре блока индексной арифметики, что повышает быстродействие ЭВМ, но и увеличивает объем оборудования.

Индексация служит не только для организации циклов, но в сочетании с базированием является средством описания элементов информационных массивов, размещенных в ОП. Индексация вообще позволяет достаточно легко модифицировать исполнительные адреса, оставляя неизменным код самой команды, хранящейся в ОП.

Для управления индексацией используются команды, задающие операции над содержимым индексных регистров – *команды индексной арифметики*. Основные виды индексных операций следующие:

- засылка в соответствующий индексный регистр начального значения индекса;
- изменение индекса;
- проверка окончания циклических вычислений.

Очень коротко рассмотрим последние два типа команд.

Изменение индекса состоит в сложении (вычитании) значения индекса с фиксированным приращением, производимым каждый раз при повторении цикла. Команда изменения индекса указывает номер индексного регистра, а также значение и знак (либо адрес) приращения. Если приращение всегда фиксировано и равно, например, 1 или L (длина слова в байтах), то используются команды с подразумеваемым операндом.

Для проверки окончания цикла используют или обычную команду перехода по условию, налагаемому в этом случае на какую-либо переменную, изменяющуюся в процессе вычислений, или специальные команды "УП по счетчику" (УПС) и "УП по индексу" (УПИ).

Условный переход по счетчику (УПС)

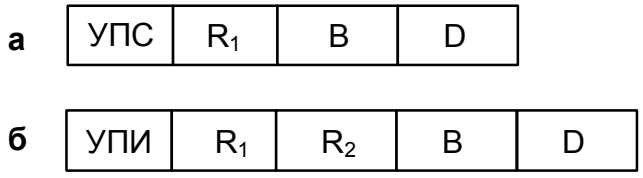
Счетчиком служит обычно один из индексных регистров, в который перед началом цикла загружается число итераций цикла. Возможная структура команды показана на рис. 5.21, а.

В данном случае счетчик организован на индексном регистре R_1. Пусть для адресации сегмента кодов команд используется базирование, тогда адрес АИ = (B) + D является адресом начала цикла. Команда УПС уменьшает содержимое регистра R_1 на 1 при каждой итерации цикла. В случае неравенства содержимого счетчика нулю ($(R_1) \neq 0$) осуществляется переход на АИ, т.е. в начало цикла. Если $(R_1) = 0$, то осуществляется переход к команде, следующей по порядку за командой УПС, т.е. выход из цикла.

Условный переход по индексу (УПИ)

Возможная структура команды показана на рис 5.21, б. Пусть R_1 – номер индексного регистра, а R_2 – номер регистра, хранящего приращение. Регистр, в котором хранится предельное значение индекса, имеет подразумеваемый адрес и, следовательно, должен быть фиксированным для данного процессора. Пусть это будет регистр, следующий по порядку за регистром, хранящим приращение, т.е. R_3. Команда УПИ складывает содержимое регистров R_1 и R_2, помещает результат сложе-

ния в R_1 и сравнивает его с содержимым R_3. Если результат сложения меньше или равен содержимому R_3, то управление передается по адресу АИ = (B) + D, т.е. происходит повторение цикла. В противном случае выполняется переход к команде, следующей по порядку за командой УПИ, т.е. выход из цикла.



R_1, R_2 – номера индексных регистров;
B – номер базового регистра;
D – смещение

Рис. 5.21. Структура команды: а – УПС; б – УПИ

Специальные команды индексной арифметики имеют, как правило, аппаратную реализацию, что позволяет свести к минимуму временные затраты на модификацию исполнительных адресов команд и операндов, связанных с управлением вычислительными циклами.

Рассмотренный выше механизм индексации можно назвать классическим. В современных ЭВМ используются различные его модификации, причем во многих случаях он не присутствует в чистом виде, а реализуется через другие механизмы организации вычислительного процесса. Кроме того, следует иметь в виду, что назначение полей адресного кода (B, D, X) зависит от архитектуры ЭВМ, разрядности машинного слова, системы команд, терминологии, используемой конкретным производителем ЭВМ в технической документации.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. В чем отличие машинной команды от машинной операции? Перечислите основные группы машинных команд.
2. Опишите и изобразите структуру машинной команды.
3. Почему при проектировании ЭВМ важно выбрать оптимальную структуру и форматы команд?
4. Какими бывают структуры машинных команд?
5. В каком случае используется безадресная структура машинной команды?
6. Чем различаются способы адресации "подразумеваемый операнд" и "подразумеваемый адрес"?
7. Опишите регистровую адресацию.
8. Какие существуют варианты формирования исполнительного адреса при базировании?
9. Как происходит адресация памяти в случае с косвенной адресацией?
10. Опишите автоинкрементную/автодекрементную адресацию.
11. Приведите способы реализации стека.
12. Как происходят операции чтения и записи числа в "перевернутый стек"?
13. Что происходит в результате выполнения команды БП?
14. Опишите структуру команды УП.
15. Как выполняется команда безусловного перехода к подпрограмме?
16. Зачем нужны индексные регистры?
17. Приведите пример использования механизма индексации.
18. Для чего используется механизм автоматического изменения исполнитель ных адресов команд?
19. Перечислите типы индексных операций.
20. Какие типы команд используются для проверки окончания цикла при индексации?

КОНТРОЛЬНЫЕ ЗАДАНИЯ

1. На листах ответа должны быть указаны номер группы, фамилия студента и номер его варианта.
2. Номера вопросов выбираются студентом в соответствии с двумя последними цифрами в его зачетной книжке. В табл.5.1 a_{n-1} – это предпоследняя цифра номера, a_n – последняя цифра. В клетках таблицы стоят номера вопросов, на которые необходимо дать письменный ответ.

Номера вопросов Таблица 5.*1*

a_{n-1} \ a_n	0	1	2	3	4	5	6	7	8	9
0	1,5,9, 13,16	2,6,10, 14,17	3,7,11, 15,18	4,8,12, 13,19	1,9,11, 14,20	2,10,12, 15,18	3,5,8, 13,17	4,6,7, 14,19	1,8,10, 13,17	2,5,7, 14,18
1	3,6,8, 15,19	4,7,9, 13,16	1,5,12, 14,20	2,6,11, 15,16	3,7,10, 13,17	4,8,9, 14,18	1,7,12, 13,18	2,8,10, 14,19	3,5,11, 15,20	4,6,9, 13,17
2	2,7,9, 13,20	1,8,11, 15,19	4,5,10, 15,17	3,6,12, 14,16	1,5,9, 13,16	2,6,10, 14,17	3,7,11, 15,18	4,8,12, 13,19	1,9,11, 14,20	2,10,12, 15,18
3	3,5,8, 13,17	4,6,7, 14,19	1,8,10, 13,17	2,5,7, 14,18	3,6,8, 15,19	4,7,9, 13,16	1,5,12, 14,20	2,6,11, 15,16	3,7,10, 13,17	4,8,9, 14,18
4	1,7,12, 13,18	2,8,10, 14,19	3,5,11, 15,20	4,6,9, 13,17	2,7,9, 13,20	1,8,11, 15,19	4,5,10, 15,17	3,6,12, 14,16	1,5,9, 13,16	2,6,10, 14,17
5	3,7,11, 15,18	4,8,12, 13,19	1,9,11, 14,20	2,10,12, 15,18	3,5,8, 13,17	4,6,7, 14,19	1,8,10, 13,17	2,5,7, 14,18	3,6,8, 15,19	4,7,9, 13,16
6	1,5,12, 14,20	2,6,11, 15,16	3,7,10, 13,17	4,8,9, 14,18	1,7,12, 13,18	2,8,10, 14,19	3,5,11, 15,20	4,6,9, 13,17	2,7,9, 13,20	1,8,11, 15,19
7	4,5,10, 15,17	3,6,12, 14,16	1,5,9, 13,16	2,6,10, 14,17	3,7,11, 15,18	4,8,12, 13,19	1,9,11, 14,20	2,10,12, 15,18	3,5,8, 13,17	4,6,7, 14,19
8	1,8,10, 13,17	2,5,7, 14,18	3,6,8, 15,19	4,7,9, 13,16	1,5,12, 14,20	2,6,11, 15,16	3,7,10, 13,17	4,8,9, 14,18	1,7,12, 13,18	2,8,10, 14,19
9	3,5,11, 15,20	4,6,9, 13,17	2,7,9, 13,20	1,8,11, 15,19	4,5,10, 15,17	3,6,12, 14,16	1,5,9, 13,16	2,6,10, 14,17	3,7,11, 15,18	4,8,12, 13,19

6. ПРИНЦИПЫ ОРГАНИЗАЦИИ СИСТЕМ ПРЕРЫВАНИЯ ПРОГРАММ

В процессе выполнения программ внутри ЭВМ или во внешней среде могут возникнуть события, требующие немедленной реакции со стороны процессора. Реакция состоит в том, что процессор прерывает обработку текущей программы (прерываемой программы) и переходит к выполнению некоторой другой программы (прерывающей программы), специально предназначенной для данного события. По завершении этой программы процессор возвращается к выполнению прерванной программы. Рассматриваемый процесс называется *прерыванием программы* и может быть пояснен рис. 6.1.

Каждое событие, требующее прерывания, сопровождается сигналом, оповещающим об этом ЭВМ и называемым *запросом прерывания*. Прерывания могут порождаться внутренними и внешними событиями:

Внутренние – сбой в аппаратуре, переполнение разрядной сетки, деление на 0, выход из установленной зоны памяти, попытка обратиться к запрещенной зоне памяти, попытка обращения к защищенным программам операционной системы, сигнал от таймера и т.д.

Внешние – запрос от другой ЭВМ, сообщение от аварийных датчиков управляемого технологического процесса, запрос оператора, требование от ПУ операции обмена, запросы на обслуживание клавиатуры, мыши и т.д.

В общем случае запросы прерывания генерируются несколькими, развивающимися параллельно во времени процессами, которые в некоторый момент времени требуют вмешательства процессора. Общим во всех этих запросах является то, что моменты их поступления невозможно предусмотреть. Это существенно отличает процесс прерывания от рассмотренного ранее процесса передачи управления подпрограмме, происходящего в заранее известных точках основной программы.

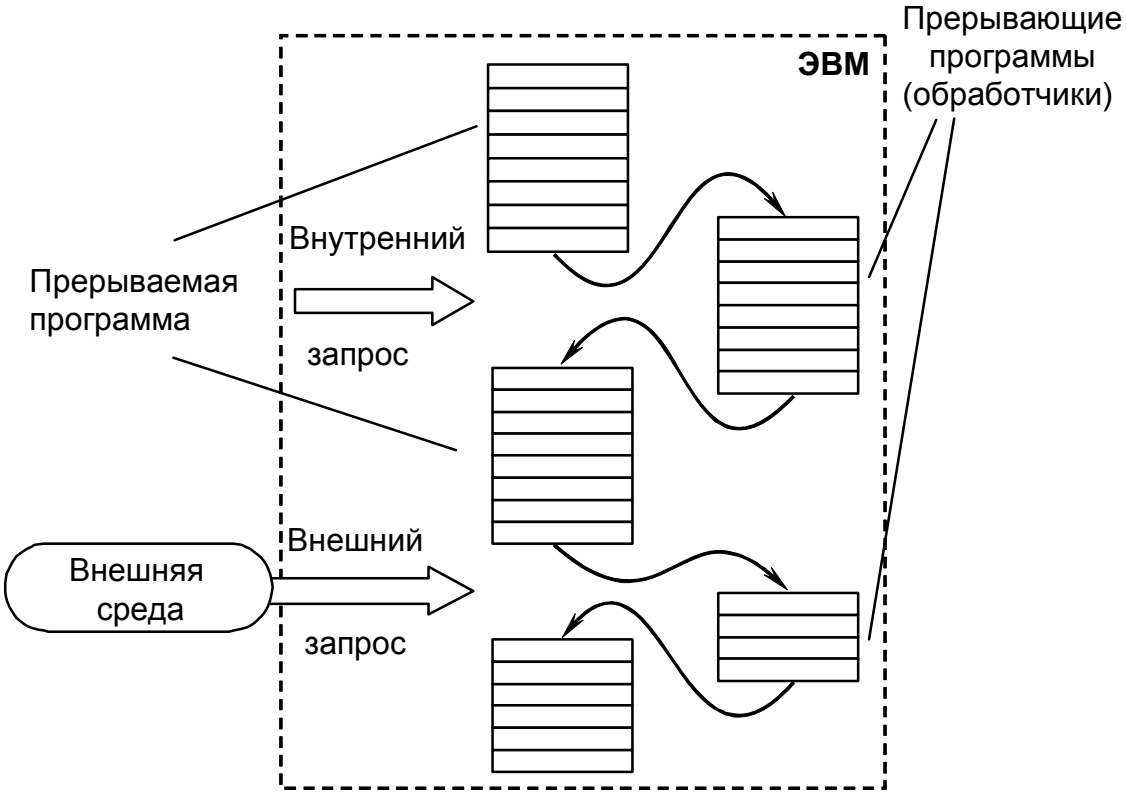


Рис. 6.1. Прерывание программы

Возможность прерывания – важное свойство ЭВМ, позволяющее эффективно использовать производительность процессора и прежде всего при организации параллельной работы процессора и периферийных устройств ЭВМ.

Для эффективной организации процесса прерывания и минимизации усилий программиста современные ЭВМ снабжены соответствующими программно-аппаратными средствами, которые получили название *контроллера прерываний*.

Контроллер прерываний в общем случае является достаточно сложным программируемым устройством, требующим соответствующей инициализации со стороны процессора. В процессе инициализации в управляющие регистры контроллера загружается информация о дисциплине обслуживания запросов прерывания, количестве используемых входов, режиме взаимодействия с процессором и т.д. Обычно инициализация контроллера выполняется при запуске вычислительной системы. Однако большинство контроллеров прерываний допускают перепрограммирование и в процессе обработки программы, в частности изменение дисциплины обслуживания поступающих запросов прерывания. Конструктивно контроллеры прерываний выполняются в виде отдельных специализированных БИС, но в ряде случаев могут быть встроены в другие устройства вычислительной системы. Простейшие контрол-

леры прерываний небольших микроЭВМ часто строятся на логических микросхемах общего назначения.

Основными функциями системы прерывания являются:

- запоминание состояния прерываемой программы и осуществление перехода к прерывающей программе;
- восстановление состояния прерванной программы и возврат к ней.

Под термином *состояние программы* (процессора), строго говоря, следует понимать совокупность состояний всех запоминающих элементов (триггеров, регистров, ячеек памяти) в соответствующий момент времени (например, после выполнения микрокоманды, команды, программы). Однако не вся эта информация искажается при переходе к другой команде или программе, поэтому из всего многообразия информации о состоянии программы (процессора) отбирают наиболее существенные ее элементы, изменяющиеся при переходе к другой команде или программе.

Вектор состояния в каждый момент времени должен содержать информацию, достаточную для запуска программы с точки, соответствующей моменту формирования данного вектора состояния. При этом предполагается, что другая информация о состоянии узлов процессора либо не существенна, либо может быть восстановлена программным путем.

Вектор состояния формируется в соответствующих регистрах процессора, изменяясь после выполнения каждой команды. Наборы информационных элементов, образующих вектор состояния, отличаются у ЭВМ разных типов и зависят от сложности процессора. В простейших процессорах эти наборы невелики. Например, в процессоре КР580ВМ80 (I8080) вектор состояния состоит из содержимого счетчика адреса команд (16 бит), содержимого регистра признаков (8 бит) и содержимого аккумулятора (8 бит). В более сложных процессорах вектор состояния может содержать существенно большее количество элементов.

Аналогично, *вектор начального состояния* должен содержать всю необходимую информацию для начального запуска программы. Во многих случаях вектор начального состояния содержит только один элемент – начальный адрес запускаемой программы.

При рассмотрении систем прерывания очень широко используется также термин *вектор прерывания*, который является ничем иным, как вектором начального состояния прерывающей программы (обработчика). Вектор прерывания содержит всю необходимую информацию для перехода к обработчику, в том числе его начальный адрес. Каждому уровню прерываний (см. п. 6.1), а в простых ЭВМ каждому входу прерывания (периферийному устройству) соответствует свой вектор прерывания, который инициализирует выполнение соответствующего обработчика. Обычно векторы прерывания хранятся в специально выделенных фиксированных ячейках памяти с короткими адресами. Таким образом, для перехода к соответствующей прерывающей программе процессор должен располагать не только вектором прерывания, но и адресом этого вектора.

Следует иметь в виду, что понятие вектор прерывания достаточно условно, поскольку в абсолютном большинстве случаев вектор прерывания состоит только из одного элемента – начального адреса прерывающей программы (обработчика).

При наличии нескольких источников запросов должен быть установлен определенный порядок обслуживания поступающих запросов, т.е. должны быть установлены *приоритетные соотношения (дисциплина обслуживания)*. Они определяют, какой из нескольких запросов, поступивших одновременно, подлежит обработке в первую очередь, имеет ли право данный запрос прерывать ту или иную программу и т.д. Все это входит в процедуру перехода к прерывающей программе.

6.1. ХАРАКТЕРИСТИКИ СИСТЕМ ПРЕРЫВАНИЯ

Эффективность систем прерывания ЭВМ может оцениваться по весьма многочисленным характеристикам, которые отражают особенности их технической реализации. Однако для изучения общих принципов построения систем прерывания достаточно рассмотреть только самые обобщенные. К их числу относятся следующие характеристики.

- *Общее количество запросов прерывания*

Количество запросов прерывания (источников запросов прерывания – ИЗП) существенно различается у ЭВМ различных типов и может достигать десятков. В системах прерывания радиальной структуры это понятие может совпадать с понятием количества входов в систему прерывания. При цепочечной организации системы прерывания эти понятия не совпадают.

- ***Время реакции***

Время реакции определяется как временной интервал между появлением запроса прерывания и началом выполнения прерывающей программы.

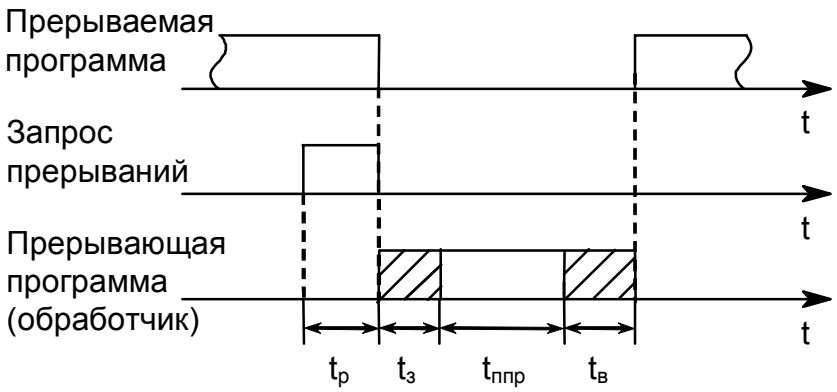
На рис. 6.2 приведена упрощенная временная диаграмма процесса прерывания в предположении, что управление запоминанием и возвратом возложено на сам обработчик. В этом случае он состоит из трех частей – подготовительной и заключительной, обеспечивающих переключение программ, и собственно прерывающей программы, выполняющей затребованные запросом операции. Кроме того, предполагается, что запрос представлен уровнем потенциала.

Для одного и того же запроса задержка в исполнении прерывающей программы (обработчика данного запроса) зависит от того, сколько запросов со старшим приоритетом ожидает обслуживания, поэтому время реакции определяется для запроса с наивысшим приоритетом. На рис. 6.2 оно обозначено $t_р$.

- *Задержка прерывания (издержка прерывания)*

Задержка прерывания ($t_{зад}$) определяется суммарным временем на запоминание ($t_з$) и восстановление ($t_в$) программы (см. рис. 6.2):

$$t_{зад} = t_з + t_в .$$



$t_р$ – время реакции системы на прерывание;
$t_з$ – время запоминания состояния прерываемой программы;
$t_{ппр}$ – время собственно прерывающей программы;
$t_в$ – время восстановления состояния прерванной программы

Рис. 6.2. Упрощенная временная диаграмма процесса прерывания

- *Глубина прерывания*

Глубина прерывания определяет максимальное число программ, которые могут прерывать друг друга. Если после перехода от основной программы к прерывающей обслуживание остальных запросов запрещено, то считается, что система имеет глубину прерывания, равную 1. Глубина равна n, если допускается последовательное прерывание до n программ. Глубина прерывания обычно совпадает с числом уровней приоритетов в системе прерывания. Если глубина прерывания не равна 1, то упрощенно это можно изобразить диаграммой (рис. 6.3). Здесь имеется в виду, что приоритет прерываний возрастает у каждого следующего запроса. Системы с большим значением глубины прерывания обеспечивают более быструю реакцию на срочные запросы.

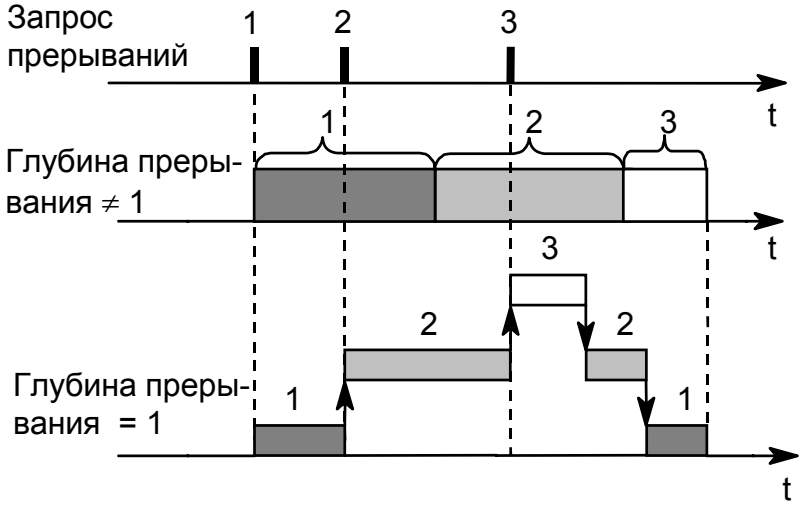


Рис.6.3. Упрощенная временная диаграмма процесса прерывания в системах с различной глубиной

- *Насыщение системы прерывания*

Если запрос окажется не обслуженным к моменту прихода нового запроса от того же источника (т.е. того же приоритета), то возникает явление, называемое *насыщением системы прерывания*. В этом случае часть запросов прерывания будет утрачена, что для нормальной работы ЭВМ недопустимо. Поэтому быстродействие ЭВМ, характеристики системы прерывания и частоты возникающих запросов должны быть строго согласованы, чтобы насыщение было невозможно.

- *Допустимые моменты прерывания программ*

В большинстве случаев прерывания допускаются после выполнения любой текущей команды, когда время реакции на прерывание определяется, в основном, длительностью выполнения одной команды.

При работе ЭВМ с быстрыми технологическими процессами в реальном масштабе времени (т.е. в контурах управления реальных физических процессов) это время может оказаться недопустимо большим. Кроме того, существуют задачи, при выполнении которых требуется немедленная реакция на ошибку, обнаруженную, например, аппаратурой контроля, чтобы не допустить выполнения ошибочно сформированного кода команды. Такие ситуации характерны для управляющих ЭВМ военного назначения.

В этом случае в системе прерывания реализуется возможность прерывания после любого такта выполнения команды программы. Однако это требует запоминания, а потом восстановления гораздо большего объема информации, чем в случае прерывания после окончания команды, поэтому такая организация прерываний воз-

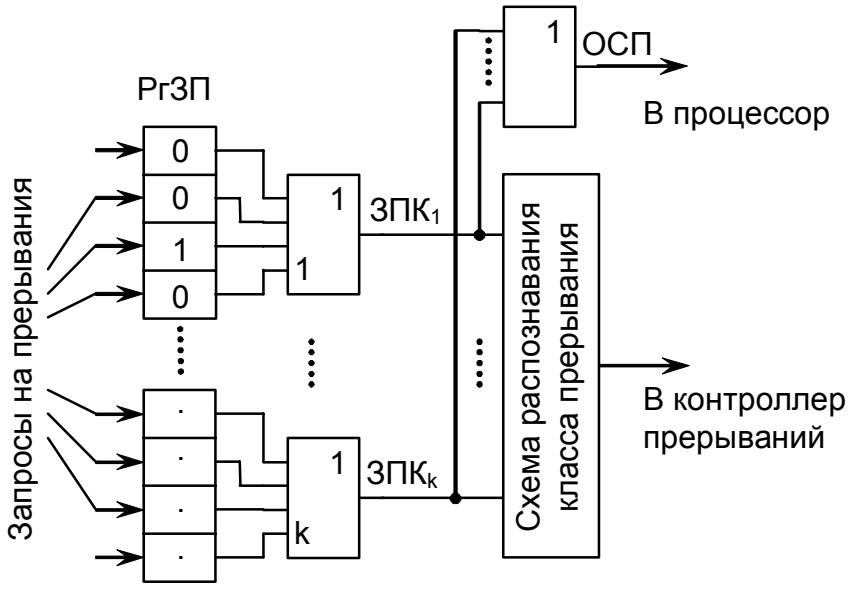
можна только в ЭВМ с быстродействующей сверхоперативной памятью достаточного объема.

- *Число уровней прерываний*

Уже отмечалось, что в ЭВМ число различных источников запросов (причин) прерывания может достигать десятков и даже сотен. Однако в ряде случаев многие запросы поступают от групп однотипных устройств, для обслуживания которых требуется одна и та же прерывающая программа (обработчик). Запросы от однотипных устройств целесообразно объединить в группы, каждой из которых будет соответствовать свой сигнал запроса прерывания.

Уровнем или *классом* прерывания называется совокупность запросов, инициирующих одну и ту же прерывающую программу (обработчик).

Технически такое объединение может быть реализовано по-разному. На рис. 6.4 приведен возможный вариант решения этой задачи при условии, что все поступающие запросы фиксируются в разрядах регистра запросов прерывания (РгЗП).



РгЗП – регистр запросов прерывания;
ОСП – общий сигнал прерывания;
ЗПК – запросы классов прерывания

Рис. 6.4. Разделение запросов на классы прерывания

Запросы от всех источников поступают в РгЗП, устанавливая соответствующие его разряды (флажки) в состояние 1, указывающее на наличие запроса прерывания. Запросы классов прерывания $ЗПК_1$-$ЗПК_k$ формируют элементы ИЛИ, объединяющие разряды РгЗП, относящиеся к соответствующим уровням. Еще одна схема ИЛИ формирует ОСП, поступающий в УУ процессора. Он формируется при любом запросе (поднятом флажке) прерывания.

Информация о действительной причине прерывания (конкретном источнике запроса), породившей запрос данного класса, содержится в коде прерывания, который отражает состояние разрядов РгЗП, относящихся к данному классу прерываний. В процессе обработки прерывания эти разряды (содержащийся в них код) подвергаются анализу. Такое объединение прерываний в классы уменьшает объем аппаратуры, но замедляет работу системы прерывания. После передачи управления прерывающей программе соответствующий триггер РгЗП сбрасывается.

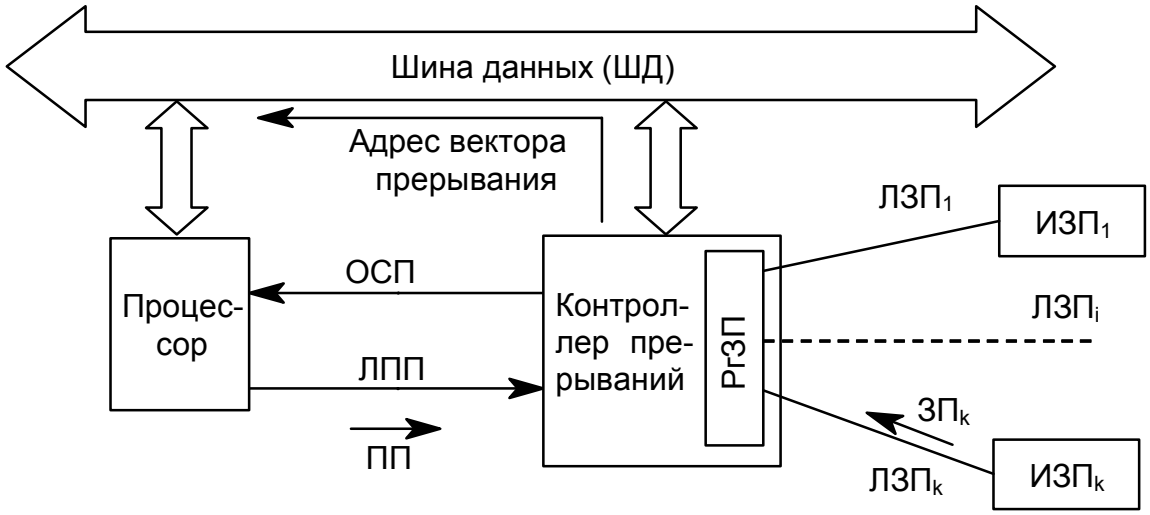
6.2. ВОЗМОЖНЫЕ СТРУКТУРЫ СИСТЕМ ПРЕРЫВАНИЯ

Конкретные технические реализации систем прерывания имеют множество вариантов и зависят от типа используемого процессора, структуры системного интерфейса, целевого назначения ЭВМ. В то же время все они являются сложными комбинациями небольшого количества базовых структур систем прерывания. Ниже рассматриваются две основные базовые структуры – радиальная и цепочечная.

- *Радиальная структура*

Обобщенная структура системы прерывания радиального типа представлена на рис. 6.5 (ША и ШУ не изображены).

Характерной особенностью радиальной структуры является то, что каждый ИЗП (в частном случае ПУ) подключен через ЛЗП к отдельному входу контроллера прерываний. В этом случае число подключаемых ИЗП определяется числом входов в систему прерывания. Поступающие запросы, как правило, фиксируются в разрядах РгЗП. Запросы от ИЗП могут быть представлены как уровнем потенциала, так и его перепадом, поскольку поступают в контроллер по отдельным линиям. Однако представление запроса потенциалом более предпочтительно, поскольку система прерывания становится более устойчивой как к помехам, так и к сбоям аппаратуры. Это существенно снижает вероятность пропуска запроса от ИЗП.



ИЗП – источник запросов прерывания;
ЗП – сигнал запроса прерывания;
ПП – сигнал подтверждения прерывания;
ОСП – общий сигнал прерывания;
ЛЗП – линия запросов прерывания;
ЛПП – линия подтверждения прерывания;
РгЗП – регистр запросов прерывания

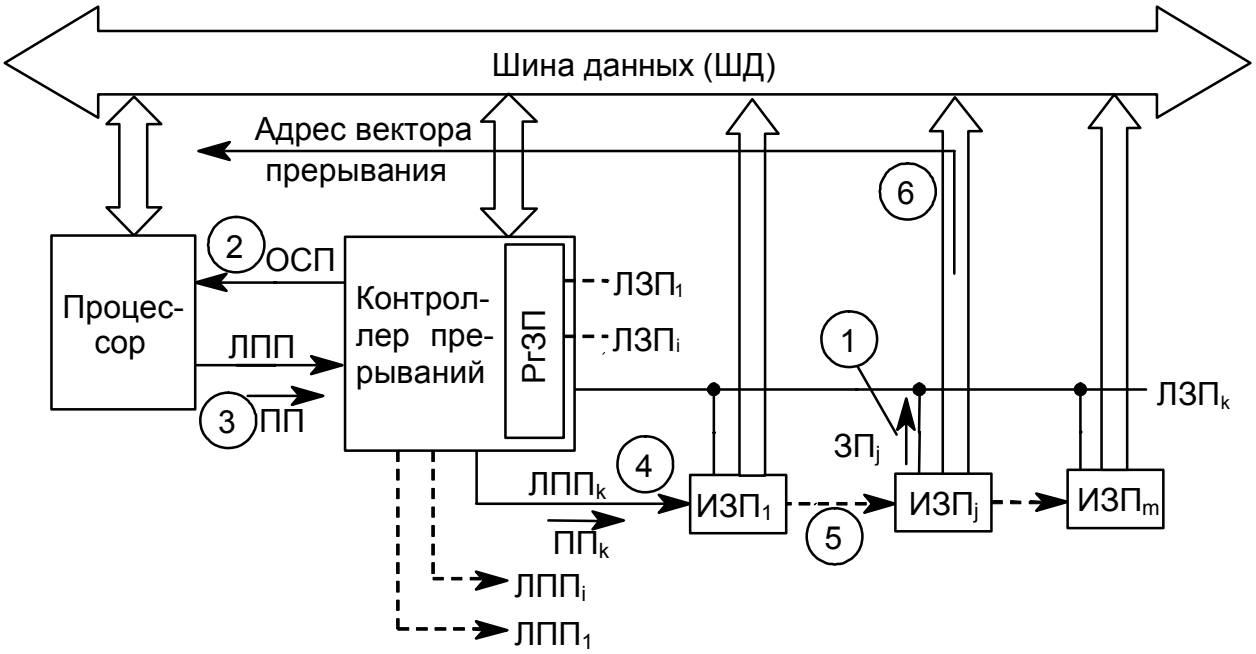
Рис. 6.5. Радиальная структура системы прерывания

Основным преимуществом такой структуры является упрощение аппаратных средств ИЗП, относящихся к обслуживанию процесса прерывания, поскольку в их функции входит только формирование сигнала запроса прерывания, т.е. необходим только запросчик.

- *Цепочечная структура*

Обобщенная структура системы прерывания цепочечного типа представлена на рис. 6.6 (ША и ШУ не изображены).

Характерной особенностью цепочечной структуры является то, что к одной ЛЗП может подключаться множество ИЗП. Каждая ЛЗП соответствует одному входу в систему прерывания и обладает собственным уровнем приоритета. В отличие от радиальной структуры запросы от ИЗП всегда представлены уровнем потенциала. Поэтому выходные каскады аппаратных средств формирования запросов в каждом ИЗП представляют собой ключи с открытым коллектором, объединенные по схеме монтажного "или". Это позволяет исключить потерю запросов, одновременно выставленных разными источниками на одну ЛЗП. Запросы прерывания, поступающие с разных ЛЗП и имеющие различный приоритет, могут фиксироваться (как и в предыдущем случае) в разрядах РгЗП.



ИЗП – источник запросов прерывания;
ЗП – сигнал запроса прерывания;
ПП – сигнал подтверждения прерывания;
ОСП – общий сигнал прерывания;
ЛЗП – линия запросов прерывания;
ЛПП – линия подтверждения прерывания

Рис. 6.6. Цепочечная структура системы прерывания
(цифрами обозначена последовательность прохождения сигналов)

Основным преимуществом такой структуры является практически неограниченное количество ИЗП, подключаемых к одному входу системы прерывания (одной ЛЗП) без снижения быстродействия. Однако сложность и объем аппаратуры поддержки системы прерывания в каждом ИЗП существенно увеличиваются, что увеличивает стоимость системы прерывания и ЭВМ в целом. Помимо запросчика каждый ИЗП должен содержать также аппаратуру формирования кода адреса вектора прерывания. Да и аппаратура самого запросчика также усложняется. Кроме того, необходимо принять меры для того, чтобы при отсутствии любого ИЗП цепь распространения сигнала ПП не размыкалась. Это обеспечивается либо специальной конструкцией контактов слота, либо перемычками на системной плате.

Несмотря на усложнение и удорожание подобных систем прерывания по сравнению с радиальными, они очень широко используются в вычислительных системах самой разной архитектуры и назначения.

6.3. ОРГАНИЗАЦИЯ ПЕРЕХОДА К ПРЕРЫВАЮЩЕЙ ПРОГРАММЕ

Конкретные реализации процедур перехода к прерывающей программе во многом зависят от структуры системы прерывания и типа используемого процессора. Между тем можно сформулировать некоторые общие принципы построения этих процедур. В любой процедуре перехода к обработчику главное место занимает передача текущего вектора состояния прерываемой программы на хранение в стек, а затем загрузка в регистры процессора вектора прерывания, т.е. начального вектора состояния обработчика, который, как уже отмечалось, в большинстве случаев состоит только из одного элемента – начального адреса. Векторы прерывания хранятся в фиксированных ячейках ОП, поэтому процессор должен предварительно получить адрес соответствующего вектора прерывания. При наличии нескольких ИЗП процедура перехода к прерывающей программе включает в себя также операции по выявлению запроса прерывания с максимальным приоритетом.

Следует отметить, что различают абсолютный и относительный приоритеты.

Абсолютный приоритет – запрос, имеющий абсолютный приоритет, при поступлении сразу же прерывает выполняемую в процессоре программу и инициирует выполнение соответствующего обработчика.

Относительный приоритет – запрос, имеющий относительный приоритет, при поступлении является первым кандидатом на обслуживание после завершения выполнения текущей программы.

Именно метод обнаружения запроса с максимальным приоритетом и способ формирования адреса вектора прерывания различают процедуры обработки прерываний в системах радиальной и цепочечной структуры. Ниже коротко рассматриваются основные моменты перехода к прерывающей программе для обоих случаев.

- **Радиальная структура**

В соответствии с рис. 6.5 все запросы от ИЗП фиксируются в разрядах РгЗП, который еще называется регистром флажков. При поступлении запроса в соответствующий разряд РгЗП записывается 1 (поднимается флажок). Дальнейшая последовательность операций по реализации процедуры перехода к прерывающей программе следующая.

1. При поступлении любого запроса (или нескольких запросов) в РгЗП формируется сигнал ОСП. Этот сигнал транслируется в процессор, а также инициирует процедуру поиска запроса с максимальным приоритетом, результатом выполнения которой является формирование кода адреса соответствующего вектора прерывания. Эта процедура может выполняться как программно, так и аппаратно. Некоторые варианты ее реализации рассматриваются ниже.
2. Процессор заканчивает выполнение текущей команды и посылает в контроллер сигнал "Подтверждение прерывания" (см. рис. 6.5).
3. В ответ на сигнал "Подтверждение прерывания" контроллер выставляет на ШД код адреса вектора прерывания.
4. Процессор считывает с ШД код адреса вектора прерывания, сохраняет в стеке вектор текущего состояния прерываемой программы (в простейшем случае только содержимое счетчика адреса команд) и осуществляет переход к прерывающей программе. Если вектор текущего состояния прерываемой программы автоматически сохраняется только частично, то операции по сохранению его остальных элементов возлагаются на обработчик (см. рис. 6.2, интервал времени tз).

- *Цепочечная структура*

В соответствии с рис. 6.6 к каждой ЛЗП (входу системы прерывания) может быть подключено множество запросчиков ИЗП, объединенных по схеме монтажное "или". Сигнал "подтверждение прерывания" распространяется по цепочке ИЗП, подключенных к одной ЛЗП. Распространение этого сигнала блокируется $ИЗП_j$, выставившим запрос. Получив сигнал "Подтверждение прерывания", $ИЗП_j$ выставляет на ШД код адреса вектора прерывания. Таким образом, приоритет подключенных к одной ЛЗП устройств определяется положением ИЗП в цепочке распространения сигнала "Подтверждение прерывания". Это исключает необходимость выполнения процедуры поиска запроса с максимальным приоритетом среди ИЗП, подключенных к одной ЛЗП.

Если ЛЗП несколько, что обычно имеет место в реальных системах прерывания цепочечной структуры, в контроллере прерывания выполняется процедура поиска возбужденной ЛЗП с максимальным приоритетом, которая аналогична процедуре поиска запроса с максимальным приоритетом в системе радиальной структуры.

На основании вышеизложенного можно сформулировать последовательность основных операций по реализации процедуры перехода к прерывающей программе. При этом предполагается, что состояние возбужденных ЛЗП фиксируется в разрядах РгЗП, несмотря на то, что при потенциальном способе задания сигналов запросов прерывания его наличие не обязательно.

1. При поступлении любого запроса (или нескольких запросов) в РгЗП формируется сигнал ОСП. Этот сигнал транслируется в процессор, а также инициирует процедуру поиска возбужденной ЛЗП с максимальным приоритетом, результатом выполнения которой является выбор линии распространения сигнала "Подтверждение прерывания" (цепочки ИЗП).
2. Процессор заканчивает выполнение текущей команды и посылает в контроллер сигнал "Подтверждение прерывания" (см. рис. 6.6).
3. Получив этот сигнал, контроллер транслирует его на выбранную цепочку ИЗП.
4. ИЗП, имеющий максимальный приоритет среди устройств, выставивших запрос на выбранную ЛЗП, блокирует дальнейшее распространение сигнала "Подтверждение прерывания" и выставляет на ШД код адреса своего вектора прерывания.
5. Процессор считывает с ШД код адреса вектора прерывания, сохраняет в стеке вектор текущего состояния прерываемой программы (в простейшем случае только содержимое счетчика адреса команд) и осуществляет переход к прерывающей программе. Если вектор текущего состояния прерываемой программы автоматически сохраняется только частично, то операции по сохранению его остальных элементов возлагаются на обработчик (см. рис. 6.2, интервал времени $t_з$).

В простых системах прерывания цепочечного типа может присутствовать только одна ЛЗП. В этом случае отпадает необходимость выполнения процедуры поиска ЛЗП с максимальным приоритетом, которая, даже если выполняется аппаратными средствами, требует сравнительно больших временных затрат. Отпадает необходимость и в контроллере прерываний, поскольку сигнал с ЛЗП может непосредственно поступать на соответствующий вход процессора вместо сигнала ОСП. Такие системы прерывания являются наиболее динамичными даже при достаточно большом количестве ИЗП.

Уже неоднократно отмечалось, что обязательным компонентом процедуры перехода к обработчику является операция обнаружения наиболее приоритетного запроса прерывания (радиальная структура) или ЛЗП (цепочечная структура), которая в большинстве случаев выполняется в контроллере прерываний. Тем самым реализуется система приоритетных соотношений между ИЗП или цепочками ИЗП. Эта система приоритетных соотношений может быть либо фиксированной, либо изменяться программным путем в процессе обработки задачи. В соответствии с этим

различаются системы прерываний с фиксированными приоритетами и с программно-управляемыми. Большинство контроллеров прерываний являются программируемыми и могут реализовывать систему как фиксированных, так и программно-управляемых приоритетов. Ниже рассматриваются некоторые варианты реализации процедур установления приоритетных соотношений обоих типов.

6.3.1. РЕАЛИЗАЦИЯ ФИКСИРОВАННЫХ ПРИОРИТЕТОВ

Рассмотрим только простейший случай установки приоритетных соотношений, состоящий в том, что уровень приоритета определяется порядком присоединения ЛЗП к входам системы прерывания, т.е. разрядам РгЗП. Пусть максимальный приоритет имеет вход с минимальным номером. В этом случае поиск ЛЗП с максимальным приоритетом сводится к поиску крайнего левого поднятого флажка в РгЗП. Процедуру поиска можно реализовать как программными, так и аппаратными средствами. В ряде случае эту процедуру называют соответственно программным или аппаратным *полингом*.

- *Программный опрос РгЗП*

Начало поиска инициируется сигналом ОСП. Программа поиска состоит из последовательности условных операторов, анализирующих разряды РгЗП, начиная с младшего. При обнаружении поднятого флажка управление передается обработчику. Если сигнал ОСП после выполнения обработчика не снимается, происходит повторный анализ разрядов РгЗП и поиск ЗП с максимальным приоритетом среди ЗП, оставшихся не обслуженными.

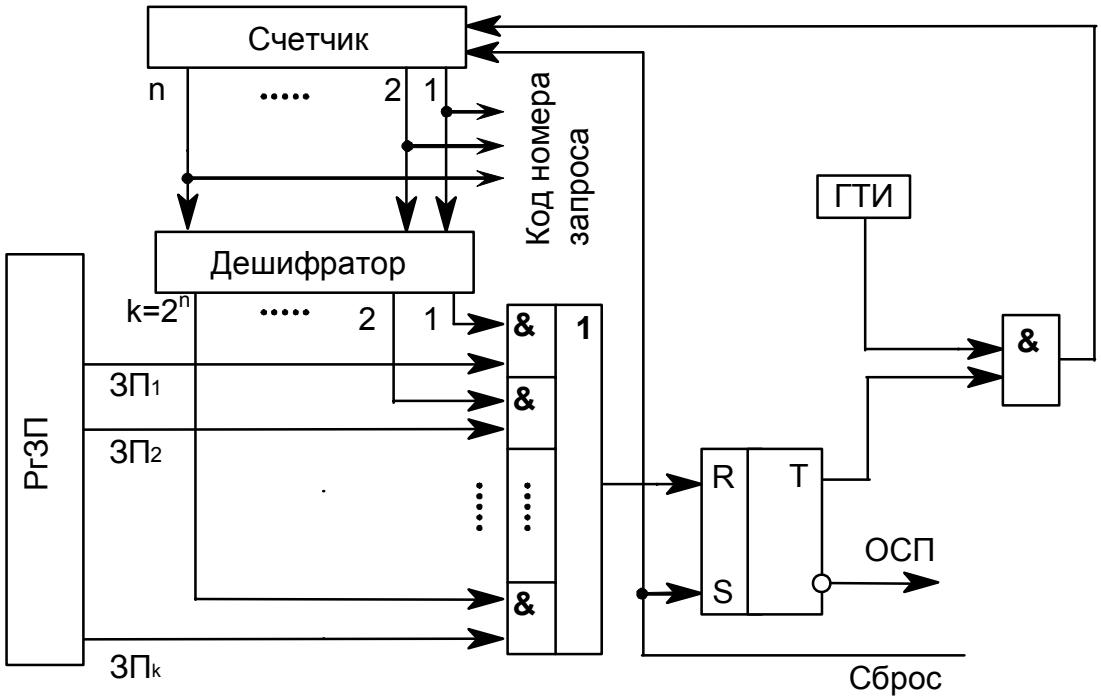
Программный полинг требует минимальных аппаратных затрат, поскольку выполняется процессором, а в качестве РгЗП можно использовать какой-либо управляемый регистр (т.е. БИС контроллера прерываний не требуется). Однако существенным недостатком метода является большое время поиска ЗП, особенно при большом количестве ЛЗП.

Для уменьшения времени поиска ЗП с максимальным приоритетом используют аппаратную реализацию алгоритма. Ниже рассматривается возможная структура такого устройства.

- *Аппаратный циклический опрос РгЗП*

Структурная схема устройства, реализующего аппаратный циклический опрос разрядов РгЗП, представлена на рис. 6.7. Устройство аппаратного полинга входит в состав контроллера прерываний и функционирует следующим образом.

Опрос k линий запросов прерываний производится с помощью n-разрядного счетчика ($2^n \geq k$), на вход которого поступают импульсы от ГТИ. Поиск начинается со сброса счетчика и установки Т в состояние 0. При этом импульсы от ГТИ через схему & начинают поступать на вход счетчика. На выходах дешифратора, начиная с первого, последовательно появляются единицы. Это происходит до тех пор, пока не попадается возбужденная линия запроса прерывания, на которой тоже выставлена 1. В этом случае триггер Т перебрасывается в состояние 1 и в процессор идет сигнал ОСП. Кроме того, прекращается поступление импульсов ГТИ на вход счетчика, т.е. завершается просмотр входов системы прерывания. Содержимое счетчика – код номера запроса (старшего по приоритету) используется для формирования адреса соответствующего вектора прерывания. После передачи управления обработчику счетчик и триггер сбрасываются в 0 сигналом "Сброс", и процедура опроса разрядов РгЗП возобновляется.



РгЗП – регистр запросов прерываний;
ЗП – запрос на прерывание;
ОСП – общий сигнал запроса на прерывание;
ГТИ – генератор тактовых импульсов

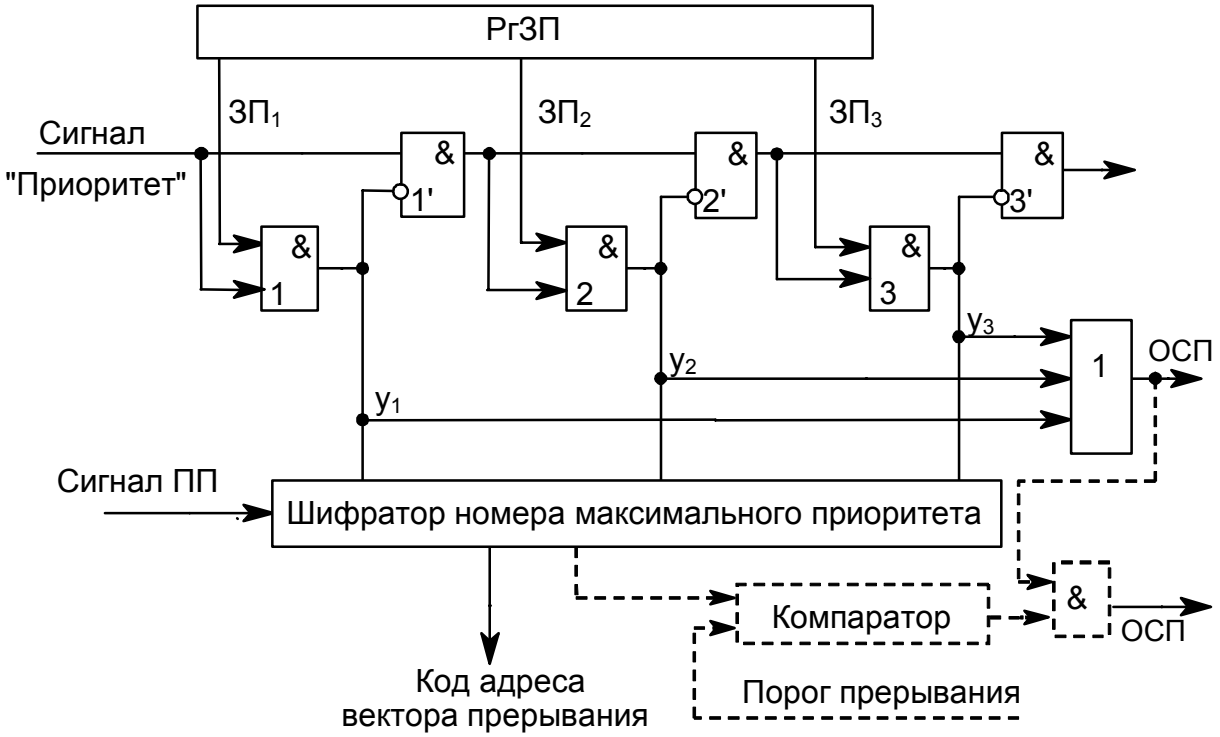
Рис. 6.7. Схема устройства циклического опроса РгЗП

Рассмотренный метод опроса разрядов РгЗП существенно быстрее программного, но при большом числе ЛЗП время реакции системы оказывается достаточно большим.

- *Аппаратный однотактный опрос РгЗП*

Структурная схема устройства, реализующего аппаратный однотактный опрос разрядов РгЗП, представлена на рис. 6.8. Схемы подобного типа принято называть *дейзи-цепочками*, и в настоящее время они получили очень широкое распространение в различных устройствах ЭВМ. Схема функционирует следующим образом. Процедура поиска запроса с максимальным приоритетом инициируется сигналом "Приоритет", поступающим на цепочку последовательно соединенных схем &. При отсутствии запросов этот сигнал пройдет насквозь всю цепочку и сигнал ОСП не сформируется. (По-прежнему считаем, что максимальный приоритет имеет запрос с минимальным номером.) Пусть выставлен $ЗП_2$. В этом случае распространение сигнала "Приоритет" блокируется $\&_{2'}$, поскольку $y_2 = 1$. Кроме того, при обнаружении любого $ЗП_i$ формируется сигнал ОСП, поступающий в процессор, а шифратор по сигналу $y_2 = 1$ формирует код адреса второго (в общем случае i-го) вектора прерывания.

По сигналу процессора "Подтверждение прерывания" этот код передается в процессор, и происходит вызов соответствующего обработчика.



ПП – подтверждение прерывания;
ОСП – общий сигнал прерывания;
ЗП – запрос прерывания;
РгЗП – регистр запросов прерываний

Рис. 6.8. Схема устройства однотактного опроса РгЗП

- *Цепочка приоритетов*

Как уже отмечалось, в цепочечной системе прерывания используется двухуровневая система приоритетов. Это приоритет ЛЗП и приоритет конкретного устройства в цепочке ИЗП, относящихся к одной ЛЗП. Приоритет ЛЗП определяется в процессе выполнения процедуры поиска наиболее приоритетного запроса (ЛЗП) в контроллере системы прерываний как радиальной, так и цепочечной структуры. Эта процедура описана выше. Приоритет устройства в цепочке ИЗП является фиксированным и определяется положением устройства на линии распространения сигнала ПП, т.е. ИЗП фактически образуют дейзи-цепочку. Таким образом, устройство, располагающееся в слоте, ближайшем к контроллеру прерываний, автоматически получает наивысший приоритет. Приоритет остальных устройств в цепочке убывает по мере удаления их слотов от контроллера прерываний. Приоритет устройства в такой системе прерывания можно изменить только путем перемещения устройства по линии распространения сигнала ПП, т.е. только перестановкой в другой слот.

При наличии нескольких наиболее приоритетных устройств такое распределение приоритетов становится недостатком, поэтому обычно используется несколько ЛЗП (4-8 линий). Наиболее приоритетные устройства в этом случае располагаются на всех ЛЗП в слотах, ближайших к контроллеру прерываний.

Во многих случаях приоритеты между прерывающими программами не могут быть зафиксированы раз и навсегда, т.е. требуется иметь возможность программно управлять приоритетами.

6.3.2. РЕАЛИЗАЦИЯ ПРОГРАММНО-УПРАВЛЯЕМЫХ ПРИОРИТЕТОВ

Существуют три основных метода реализации в ЭВМ систем программно-управляемых приоритетов – *порог прерывания*, *маска прерывания* и *смена приоритетов*. Первый используется, в основном, в системах прерывания микроЭВМ с простыми процессорами, второй – в более сложных ЭВМ с мощными процессорами, третий используется в вычислительных системах всех классов. В ряде случаев все три метода используются совместно. Ниже коротко рассматривается суть этих методов, позволяющих программным путем изменить дисциплину обслуживания ИЗП.

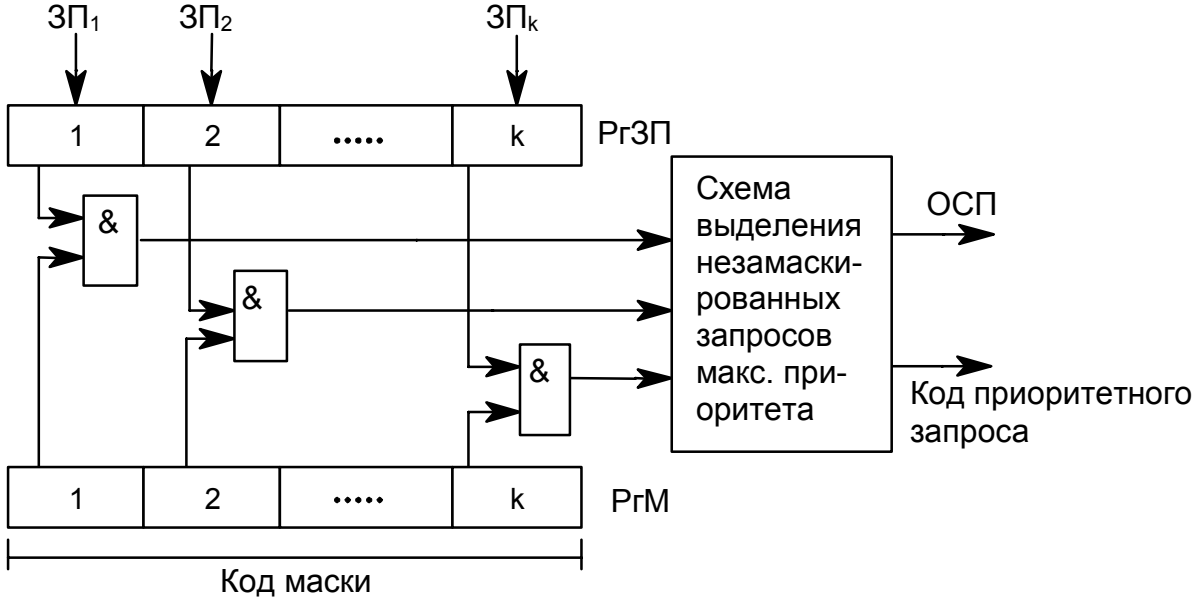
- *Порог прерывания*

Этот способ позволяет в ходе вычислительного процесса программным путем изменять уровень приоритета процессора (а следовательно, и обрабатываемой в данный момент программы) относительно приоритетов запросов источников прерывания (в основном периферийных устройств). Таким образом, устанавливается минимальный уровень приоритета запросов, которым разрешено прерывать программу, выполняемую процессором.

Порог прерывания задается командой программы, устанавливающей в регистре порога прерывания контроллера *код порога прерывания*. Вернемся к рис. 6.8, к его части, изображенной пунктиром. Ранее рассматривалось, как происходит выделение кода запроса с максимальным приоритетом. Затем этот код сравнивают с порогом прерывания (на компараторе), и, если он оказывается выше порога, вырабатывается сигнал ОСП и начинается процедура прерывания.

- *Маска прерывания*

Маска прерывания представляет двоичный код, разряды которого поставлены в соответствие запросам или классам прерываний (рис. 6.9).



ЗП – запрос на прерывание;
РгЗП – регистр запросов прерываний;
РгМ – регистр маски;
ОСП – общий сигнал прерывания

Рис. 6.9. Схема реализации управления приоритетами с помощью маски прерывания

Маска загружается командой программы в регистр маски контроллера (РгМ). Состояние 1 в данном разряде маски разрешает, а 0 запрещает (маскирует) преры-

вание текущей программы соответствующим запросом. Таким образом, программа, меняя маску в РгМ, может устанавливать произвольные приоритетные соотношения между прерывающими программами без перекоммутаций ЛЗП. Каждая прерывающая программа может устанавливать свою маску. При формировании маски 1 устанавливается в разряды, соответствующие запросам (прерывающей программе) с более высоким, чем у данной программы, приоритетом.

На изображенной схеме элементы & выделяют незамаскированные запросы прерываний, из которых специальная схема, аналогичная дейзи-цепочке, выделяет наиболее приоритетный и формирует код его номера.

С замаскированным запросом поступают в зависимости от того, какой причиной он вызван: или он игнорируется, или запоминается с тем, чтобы осуществить затребованные действия, когда запрет будет снят.

- *Смена приоритетов*

В начале п. 6.3.1 было оговорено, что разряды РгЗП, а следовательно, и входы в систему прерывания имеют фиксированные приоритеты, причем максимальный приоритет имеет вход с минимальным номером. Между тем в большинстве контроллеров прерываний имеется возможность программным путем изменять приоритеты входов, т.е. изменять дисциплину обслуживания ИЗП. Существует относительно небольшое количество используемых на практике дисциплин обслуживания. Ниже рассмотрен один из наиболее распространенных вариантов дисциплины обслуживания – режим *кругового (циклического) приоритета*.

Этот режим характерен для таких применений, в которых ИЗП (цепочки ИЗП) имеют примерно одинаковый приоритет и ни одному из них нельзя отдать предпочтение. В то же время нормальное функционирование контроллера возможно только при наличии системы приоритетов. Выравнивание приоритетов ИЗП в этом случае осуществляется следующим образом.

Каждому входу контроллера (разряду РгЗП) при инициализации присваивается соответствующий приоритет. После поступления запроса и его обслуживания (выполнения соответствующего обработчика) приоритеты входов контроллера автоматически изменяются в круговом порядке таким образом, что последний обслуженный вход получает низший приоритет. Например, контроллер прерываний имеет 4 входа, приоритеты которых после инициализации убывают в следующем порядке: $ЛЗП_1$, $ЛЗП_2$, $ЛЗП_3$, $ЛЗП_4$. Пусть в текущий момент времени завершено обслуживание $ИЗП_2$ (радиальная структура). После этого приоритеты входов автоматически будут распределены в следующем порядке: $ЛЗП_3$, $ЛЗП_4$, $ЛЗП_1$, $ЛЗП_2$. После обслуживания очередного ИЗП произойдет аналогичная смена приоритетов. Возможны и другие схемы выравнивания приоритетов ИЗП.

Одной из модификаций режима кругового приоритета является режим *адресуемого приоритета*. Этот режим аналогичен рассмотренному выше, но допускает программное определение входа контроллера, которому назначен низший или высший приоритет.

Следует иметь в виду, что современные контроллеры прерываний в большинстве случаев содержат все или почти все механизмы реализации фиксированных и программно-управляемых приоритетов, рассмотренные выше. Набор используемых дисциплин обслуживания запросов прерывания зависит от разработчиков вычислительной системы, а конкретный режим обслуживания задается программным путем.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Каково функциональное назначение систем прерывания в ЭВМ?
2. Перечислите характеристики систем прерывания. От чего зависит количество запросов прерывания?
3. Дайте определение характеристике систем прерывания "время реакции". В чем ее отличие от задержки прерывания?
4. Объясните, что понимается под характеристикой "задержка прерывания". Из чего складывается задержка прерывания?
5. Поясните смысл характеристики "глубина прерывания". Опишите явление насыщения системы прерывания.
6. В какие моменты времени допускается прерывание управляющей программы. Что понимается под классом (уровнем) прерывания? Причины деления прерываний на классы.
7. Возможные структуры систем прерывания. Опишите систему прерывания радиальной структуры.
8. Возможные структуры систем прерывания. Опишите систему прерывания с цепочечной структурой.
9. Перечислите достоинства и недостатки систем прерывания с радиальной и цепочечной структурой.
10. Опишите реализацию процедуры перехода к прерывающей программе в системе прерываний с радиальной структурой.
11. Опишите реализацию процедуры перехода к прерывающей программе в системе прерываний с цепочечной структурой.
12. Реализация прерываний с фиксированной системой приоритетов. Программный полинг.
13. Реализация прерываний с фиксированной системой приоритетов. Аппаратный циклический опрос РгЗП.
14. Опишите схему аппаратного однотактного опроса РгЗП. Что такое "дейзи-цепочка"?
15. Опишите метод реализации в ЭВМ систем программно-управляемых приоритетов – порог прерывания.
16. Приведите схему реализации управления приоритетами с помощью маски прерывания.

КОНТРОЛЬНЫЕ ЗАДАНИЯ

1. На листах ответа должны быть указаны номер группы, фамилия студента и номер его варианта.

2. Номера вопросов выбираются студентом в соответствии с двумя последними цифрами в его зачетной книжке. В табл.6.1 a_{n-1} – это предпоследняя цифра номера, a_n – последняя цифра. В клетках таблицы стоят номера вопросов, на которые необходимо дать письменный ответ.

Номера вопросов Таблица 6.1

a_{n-1} \ a_n	0	1	2	3	4	5	6	7	8	9
0	1,7, 10,13	2,8, 11,14	3,9, 12,15	4,7, 10,16	5,8, 11,13	6,9, 12,14	1,8, 10,15	2,9, 11,16	3,7, 12,13	4,8, 10,14
1	5,9, 11,15	6,7, 12,16	2,7, 10,15	1,9, 10,13	2,7, 11,14	3,8, 12,15	4,9, 10,16	5,7, 11,13	6,8, 12,14	5,9, 12,16
2	1,7, 10,13	2,8, 11,14	3,9, 12,15	4,7, 10,16	5,8, 11,13	6,9, 12,14	1,8, 10,15	2,9, 11,16	3,7, 12,13	4,8, 10,14

окончание табл.6.1

3	5,9, 11,15	6,7, 12,16	2,7, 10,15	1,9, 10,13	2,7, 11,14	3,8, 12,15	4,9, 10,16	5,7, 11,13	6,8, 12,14	5,9, 12,16
4	1,7, 10,13	2,8, 11,14	3,9, 12,15	4,7, 10,16	5,8, 11,13	6,9, 12,14	1,8, 10,15	2,9, 11,16	3,7, 12,13	4,8, 10,14
5	5,9, 11,15	6,7, 12,16	2,7, 10,15	1,9, 10,13	2,7, 11,14	3,8, 12,15	4,9, 10,16	5,7, 11,13	6,8, 12,14	5,9, 12,16
6	1,7, 10,13	2,8, 11,14	3,9, 12,15	4,7, 10,16	5,8, 11,13	6,9, 12,14	1,8, 10,15	2,9, 11,16	3,7, 12,13	4,8, 10,14
7	5,9, 11,15	6,7, 12,16	2,7, 10,15	1,9, 10,13	2,7, 11,14	3,8, 12,15	4,9, 10,16	5,7, 11,13	6,8, 12,14	5,9, 12,16
8	1,7, 10,13	2,8, 11,14	3,9, 12,15	4,7, 10,16	5,8, 11,13	6,9, 12,14	1,8, 10,15	2,9, 11,16	3,7, 12,13	4,8, 10,14
9	5,9, 11,15	6,7, 12,16	2,7, 10,15	1,9, 10,13	2,7, 11,14	3,8, 12,15	4,9, 10,16	5,7, 11,13	6,8, 12,14	5,9, 12,16

7. ПРОСТЕЙШАЯ МИКРОЭВМ

В зависимости от целевого назначения и используемого процессора ЭВМ существенно различаются по своим вычислительным возможностям, размерам, стоимости конструкции. Несмотря на то что принципы функционирования простейших микроЭВМ и мэйнфреймов в общем-то одинаковы, реальные структуры и алгоритмы функционирования даже небольших современных универсальных ЭВМ (таких, как персональный компьютер) очень громоздки и сложны для первоначального понимания. В связи с этими ниже рассматривается гипотетическая машина, обладающая чертами многих простейших микроЭВМ.

По определению, микроЭВМ – это законченная вычислительная система, построенная на базе микропроцессора и размещенная в одной БИС (однокристальная микроЭВМ) или нескольких БИС, установленных на одной плате (одноплатная микроЭВМ). Причем под законченной вычислительной системой следует понимать любое устройство переработки цифровой информации, включающее процессор, память и подсистему ввода/вывода информации, независимо от его целевого назначения, конструктивного исполнения и способов программирования.

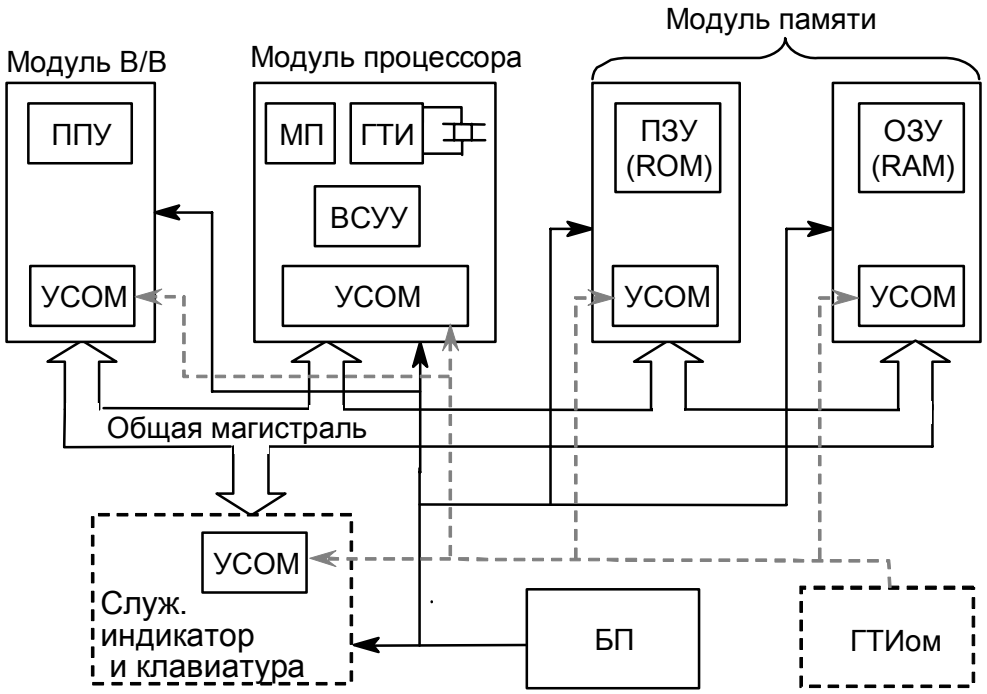
В п. 1.7 уже отмечалось, что современные микроЭВМ преимущественно имеют магистрально-модульную структуру. Ее основу в простейшем случае составляет единственная общая магистраль, к которой в необходимой номенклатуре и количестве подключены все устройства ЭВМ, выполненные в виде функционально законченных блоков. При этом все устройства ЭВМ обмениваются друг с другом информацией только через магистраль, используя схожие наборы сигналов и протоколы обмена.

Рассмотрим минимальный набор устройств, который необходим для создания простейшей гипотетической микроЭВМ (рис. 7.1).

Модуль процессора, который включает в себя непосредственно МП, генератор тактовых импульсов (ГТИ) и вспомогательные устройства управления (ВСУУ), внешние по отношению к МП.

- Модули постоянной и оперативной памяти – ПЗУ (ROM) и ОП (RAM) соответственно.
- Модуль ввода/вывода информации (В/В), содержащий приемопередающие устройства (ППУ) для связи с различными периферийными устройствами (ПУ), который может быть также назван модулем *адаптеров промежуточных интерфейсов*.

- Блок питания (БП).
- В общем случае возможно наличие модуля служебной клавиатуры и дисплея, построенного на семисегментных индикаторах или светодиодах.



ППУ – приёмопередающее устройство;
УСОМ – устройство сопряжения с общей магистралью
МП – микропроцессор;
ГТИ – генератор тактовых импульсов;
ВСУУ – вспомогательные устройства управления;
ПЗУ – постоянное запоминающее устройство;
ОЗУ – оперативно запоминающее устройство;
БП – блок питания;
ГТИом – генератор тактовых импульсов общей магистрали

Рис. 7.1. Обобщенная структура простейшей микроЭВМ

Все модули микроЭВМ имеют устройства сопряжения с общей магистралью (УСОМ). Они выполняют ряд важных функций по реализации операций обмена информацией между устройствами микроЭВМ, которые рассматриваются ниже. Следует особо подчеркнуть, что для *разных* модулей микроЭВМ структуры этих устройств и выполняемые ими функции могут быть *существенно различны*.

Одной из наиболее простых функций УСОМ является увеличение нагрузочных способностей модулей микроЭВМ и их отдельных устройств, поскольку большинство логических схем, на которых они построены, не могут обеспечить необходимую для передачи по ОМ мощность сигналов.

Кроме того, выходные (входные) каскады блоков УСОМ, присутствующих в каждом модуле микроЭВМ, имеют три устойчивых состояния:

H – высокий уровень (логическая 1);

L – низкий уровень (логический 0);

Z – состояние отключения от шины.

В состоянии Z выходное (входное) сопротивление УСОМ велико (определяется только токами утечек p-n переходов) и соответствующий модуль микроЭВМ может считаться отключенным от магистрали вообще или от ее отдельных линий.

Ранее предполагалось, что все устройства микроЭВМ синхронизируются от одного тактового генератора, расположенного в модуле процессора и обозначенного на рис. 7.1 как ГТИ. Однако, как показано на рис. 7.1, УСОМ могут (но не обязательно) синхронизироваться и от отдельного тактового генератора, обозначенного как $ГТИ_{ом}$. Индекс «ом» означает, что генератор тактовых импульсов синхронизирует операции обмена по ОМ. Возможна также синхронизация устройств ЭВМ и блоков УСОМ от одного ГТИ. В ряде случаев, как будет показано ниже, синхронизация блоков УСОМ различных модулей микроЭВМ при обмене по ОМ вообще не требуется.

В предыдущих разделах уже были рассмотрены основные принципы функционирования элементарного процессора, взаимодействие его основных частей АЛУ и УУ, а также принципы микропрограммного управления работой процессора. Рассмотрены также основные принципы построения устройств памяти, систем команд микроЭВМ и систем прерывания программ.

Теперь рассмотрим возможные варианты структур микроЭВМ, которые в большой степени определяются структурой ОМ и алгоритмами передачи сигналов по ней между различными блоками, а затем перейдем к рассмотрению непосредственно МП и принципов функционирования простейшей микроЭВМ.

7.1. СИСТЕМНЫЙ ИНТЕРФЕЙС МИКРОЭВМ. ЦИКЛ ШИНЫ

Современные процессоры конструктивно выполняются либо в виде одной БИС (СБИС), либо в виде нескольких БИС, установленных на плате модуля процессора в непосредственной близости друг от друга. Кроме того, на плате модуля процессора обычно размещается также ряд вспомогательных устройств, объединенных на схеме (рис. 7.1) в блок ВСУУ. Такими устройствами могут быть системный контроллер, контроллер прерываний, контроллер прямого доступа, таймеры и т.д. Плата модуля процессора устанавливается на общую магистраль. Это соединение физически может быть реализовано в виде разъема или запаиваемых контактов. В ряде случаев БИС МП устанавливается на магистраль непосредственно.

При взаимодействии модуля МП с модулями памяти осуществляются операции считывания или записи информации, а при взаимодействии с ПУ – операции ввода/вывода информации. При этом кроме собственно данных и адресов ячеек памяти или регистров ППУ по магистрали необходимо передавать и весьма большое количество служебных управляющих сигналов. Ввиду этого общую магистраль разделяют на три (в общем случае) самостоятельные шины:

- шину адреса (ША);
- шину данных (ШД);
- шину управления (ШУ).

Технически проще использовать однонаправленные шины, но тогда их число должно увеличиться, т.е. по две шины для операции "Чтение" (Ввод) и "Запись" (Вывод). Это приводит к существенному увеличению числа контактов разъема модуля МП или непосредственно самой БИС МП, а также числа проводников ОМ. Между тем любое увеличение числа проводников ОМ всегда приводит к увеличению стоимости ЭВМ, а в ряде случаев вообще невозможно в силу технических ограничений.

Самым очевидным способом сократить число выводов БИС и проводников ОМ является объединение однонаправленных шин в одну двунаправленную, управляемую соответствующими сигналами – запись/чтение (READ/WRITE) для модулей памяти и ввод/вывод (INPUT/OUTPUT) для модуля ППУ. Ниже рассматриваются 5 вариантов структур ОМ только с двунаправленными шинами.

- *Раздельные шины (рис.7.2, а)*

Использование отдельных двунаправленных шин упрощает обмен процессора с модулями памяти и ППУ и дает принципиальную возможность вести его в пере-

крывающиеся интервалы времени. При этом адресные пространства ячеек памяти и регистров ППУ могут перекрываться.

Основным недостатком такой структуры является большое число проводников общей магистрали и контактов модуля МП.

- *Изолированные шины (рис.7.2, б)*

Сходство процессов обмена процессор – память и процессор – регистры ППУ позволяет использовать в обоих случаях одни и те же проводники ША и ШД. Это приводит к структуре с изолированными шинами. Адресные пространства ячеек памяти и регистров ППУ, как и при использовании предыдущей структуры, могут перекрываться, т.е. они изолированы. Для того чтобы занять шины для обмена с памятью, процессор выдает сигналы READ/WRITE, а для обмена с ПУ – INPUT/OUTPUT.

По сравнению с предыдущей структурой число проводников ОМ (как и модуля МП) уменьшилось, но исчезла принципиальная возможность вести параллельный обмен с памятью и ПУ.

- *Изолированные шины и мультиплексирование ША и ШД (рис. 7.2, в)*

В этом случае ША и ШД совмещены. Вследствие этого передача адресов и данных идет в разные моменты времени. Адресные пространства ячеек памяти и регистров ППУ изолированы.

По сравнению с предыдущими структурами уменьшилось число проводников общей магистрали и выводов модуля МП, но адреса и данные могут передаваться только в неперекрывающиеся моменты времени. Это затрудняет возможность конвейеризации процесса выполнения команд и удлиняет цикл обмена процессор – память.

- *Общие шины (рис. 7.2, г)*

В данном случае команды ввода/вывода (INPUT/OUTPUT) вообще исключены, что упрощает структуру модуля МП и общей магистрали, хотя количество проводников примерно соответствует структуре с изолированными шинами. Ячейки памяти и регистры ППУ лежат в общем адресном пространстве, и для обращения к ним используются одни и те же команды.

В ряде случаев это является преимуществом, однако при возникновении определенных сбоев в работе ПУ и их некорректной обработки со стороны операционной системы возможны "зависания" вычислительного процесса.

- *Общие шины и мультиплексирование ША и ШД (рис. 7.2, д)*

Недостатки и преимущества данной структуры по сравнению с предыдущей ("общие шины") аналогичны изложенным выше для структуры, приведенной на рис. 7.2, в.

Современные МП, практически все, имеют команды ввода/вывода, т.е. дают возможность организовать структуру с изолированными шинами. При этом все они допускают обращение к регистрам ППУ как к ячейкам памяти, т.е. позволяют реализовать структуру с общими шинами.

Следует отметить, что структура магистрали типа "общие шины" является весьма распространенной в реальных устройствах. Понятия "общая магистраль (ОМ)" и "общая шина (ОШ)" в литературе часто используются как синонимы, хотя согласно приведенной выше классификации ОШ является частным случаем структуры ОМ. Ниже, при изложении материала, понятия ОШ и ОМ также будут использоваться как синонимы, за исключением особо оговоренных случаев.

Рассмотренные структуры ОМ во многом определяют внутреннюю структуру конкретной микроЭВМ. Однако структура микроЭВМ определяется также и множеством вопросов, связанных с формой представления информации и способами ее передачи внутри микроЭВМ, алгоритмами взаимодействия отдельных модулей.

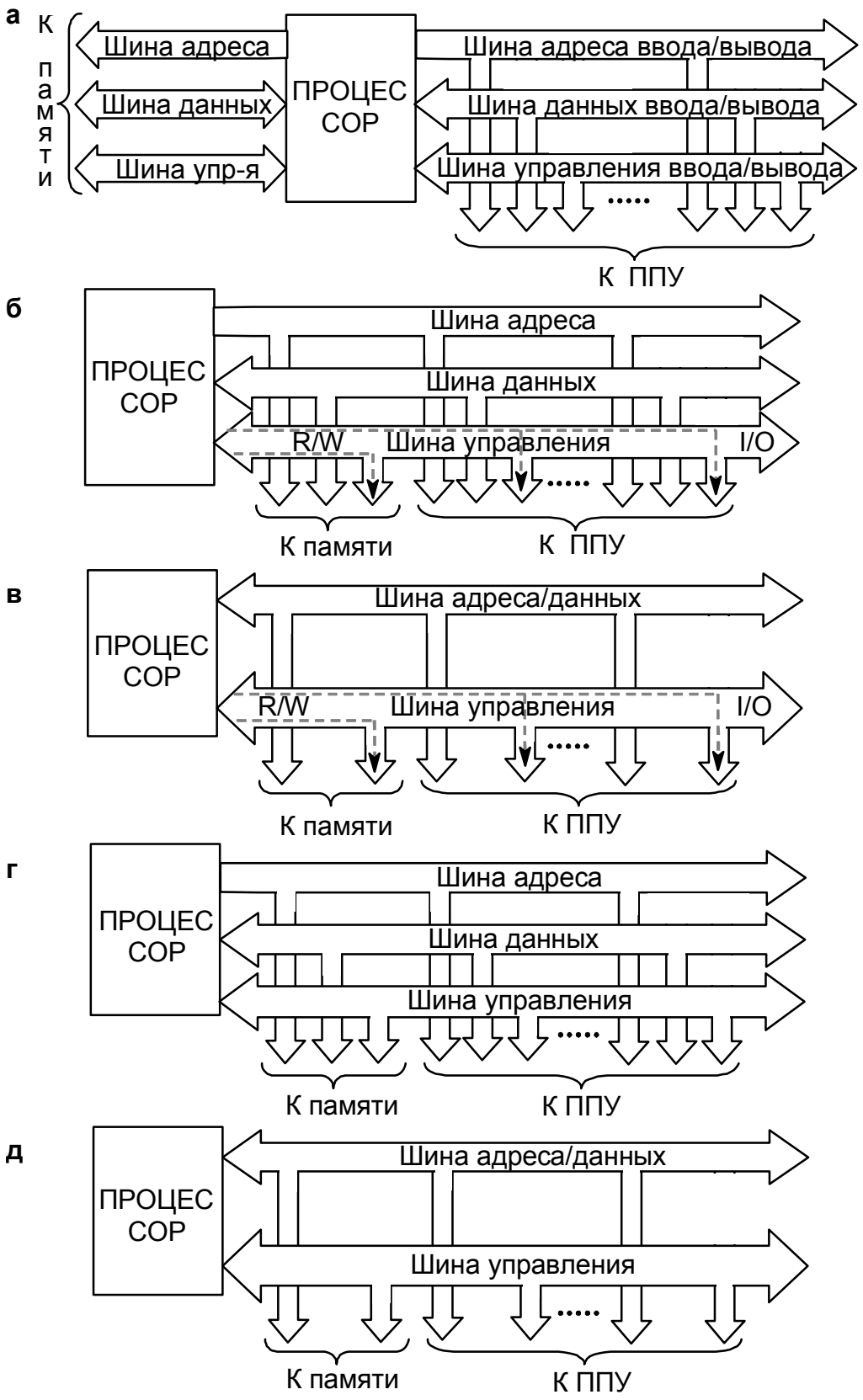


Рис. 7.2. Структуры микро-ЭВМ:
а – с раздельными шинами; б – с изолированными шинами;
в – с изолированными шинами и мультиплексированием шин адресов и данных; г – с общими шинами; д – с общими шинами и мультиплексированием шин адресов и данных

Для всесторонней характеристики структуры микроЭВМ используют весьма объемное понятие «*системный интерфейс*», включающее в себя все отмеченные выше вопросы.

Системный интерфейс – это набор цепей, связывающих процессор с памятью и ППУ, алгоритмы передачи сигналов по этим цепям, их электрические и временные параметры, тип соединительных элементов, конструктивные решения и т. д. (т.е. это комплекс аппаратно-программных средств).

Таким образом, ОМ является важной, но не единственной компонентой системного интерфейса, определяющего структуру микроЭВМ в целом.

Необходимо отметить, что в литературе редко используется термин «системный интерфейс» при описании структуры ЭВМ. Обычно используются более короткие термины «шина» или «магистраль».

Существует множество типов системных интерфейсов, разработанных для ЭВМ самых разных назначений. Количество только *стандартных* системных интерфейсов исчисляется десятками. Полное описание системного интерфейса даже одной, конкретной, ЭВМ далеко выходит за рамки настоящего курса. Между тем необходимо хотя бы коротко остановиться на другой важнейшей компоненте системного интерфейса – алгоритмах передачи сигналов по ОМ. Точнее, на основных принципах построения этих алгоритмов, поскольку они также сильно влияют на внутреннюю структуру и общие характеристики микроЭВМ. Для этого необходимо прежде всего ввести понятие «*цикл шины*».

Как уже отмечалось, в ЭВМ магистрально-модульной архитектуры наличие единого ресурса (магистрали) позволяет вести обмен между устройствами только в не перекрывающиеся моменты времени. Это означает, что в каждый момент времени существует только *один* канал связи между *двумя* устройствами, которые могут быть условно названы «передатчик» и «приемник». Возможны различные процедуры обмена по ОМ. Это запись в ОП, считывание из ОП, запись в регистры ППУ, считывание из регистров ППУ, прямой доступ к памяти, а также всевозможные модификации указанных операций. Конкретный протокол, по которому ведется обмен между двумя устройствами ЭВМ, всегда соответствует типу выполняемой процедуры. При выполнении операций обмена УСОМ передатчика выставляет на линии магистрали подлежащую передаче информацию. УСОМ приемника, получив соответствующие управляющие сигналы, должен ее считать. Между моментом установки данных на линиях магистрали и их считыванием возникает некоторый временной интервал, величина которого в общем случае может изменяться. Это обусловлено прежде всего особенностями алгоритма взаимодействия конкретных передатчика и приемника, а также тем, что помимо передачи собственно данных формируется ряд служебных управляющих сигналов, которые необходимы для реализации протокола обмена, причем их количество и номенклатура могут быть различны. Операции блоков УСОМ передатчика и приемника по реализации процедуры обмена могут, но не обязательно, синхронизироваться импульсными последовательностями от ГТИ или $ГТИ_{ом}$ (в дальнейшем – «синхроимпульсы магистрали»). Синхронизация операций обмена, если она присутствует, может осуществляться как фронтами, так и уровнями синхроимпульсов магистрали. При выполнении различных процедур обмена между устройствами микроЭВМ количество и номенклатура служебных операций, а следовательно, и интервалы времени между операциями установления данных на линиях ОМ и их считыванием могут существенно различаться. Тем более продолжительности операций обмена различаются в микроЭВМ, использующих различные типы системных интерфейсов.

Цикл шины (магистрали) – это совокупность служебных операций блоков УСОМ передатчика и приемника, необходимых для реализации конкретной процедуры обмена по ОМ между двумя устройствами ЭВМ.

Для конкретной ОМ существуют различные циклы шины, которые носят соответствующие названия по типам реализуемых процедур обмена. Это шинный цикл чтения регистра ППУ (порта), шинный цикл чтения ячейки памяти, шинный цикл записи в ячейку памяти и т. д.

Длительность цикла шины (магистрали) – это интервал времени, необходимый для реализации конкретной однократной процедуры обмена по ОМ между двумя устройствами ЭВМ. При наличии синхронизации операций обмена этот интервал может измеряться количеством необходимых синхроимпульсов магистрали (количеством тактов шины).

Как уже отмечалось, конкретные количество, номенклатура и последовательность выполнения служебных операций (структура цикла), а также их продолжительность для разных процедур обмена могут существенно различаться. Между тем *основные принципы* построения протоколов различных процедур обмена для конкретной ОМ *одинаковы*. В общем случае выделяют 4 основных типа протоколов обмена (обычно говорят «типа циклов»), каждый из которых определяет тип ОМ используемой в конкретной ЭВМ и особенности ее внутренней структуры:

- синхронный цикл (синхронные магистрали);
- асинхронный цикл (асинхронные магистрали);
- замкнутый цикл (замкнутые магистрали);
- разомкнутый цикл (разомкнутые магистрали).

Рассмотрим перечисленные варианты магистралей несколько подробнее, учитывая тот факт, что данная классификация характеризует принципы функционирования ОМ в разных аспектах. Первые два пункта учитывают наличие синхронизации при выполнении операций обмена. Последние два – наличие информационной обратной связи между передатчиком и приемником. Это означает, что и *синхронная*, и *асинхронная* магистраль может быть как *замкнутой*, так и *разомкнутой*.

- *Синхронные магистрали*

Отличительной чертой магистралей этого типа является наличие строгой привязки всех операций по реализации цикла обмена к фронтам или уровням синхроимпульсов магистрали.

Основным преимуществом синхронных магистралей является то, что они имеют более простую логику управляющих устройств блоков УСОМ и обеспечивают наивысшую пропускную способность при обмене. Основным недостатком синхронных магистралей является то, что они требуют комплексную синхронизацию блоков УСОМ, дополнительное оборудование и программное обеспечение, а также примерно одинаковое быстродействие всех устройств магистрали.

- *Асинхронные магистрали*

Отличительной чертой магистралей этого типа является отсутствие какой-либо синхронизации операций по реализации цикла обмена, т.е. $ГТИ_{ом}$ отсутствует.

Основным преимуществом асинхронных магистралей является то, что они обладают повышенной гибкостью и позволяют связывать в единую систему устройства ЭВМ, имеющие различное быстродействие. Это свойство оказывается очень важным при построении открытых управляющих систем, например систем АСУ ТП. Основной недостаток асинхронных магистралей в ограниченной пропускной способности при обмене данными. Кроме того, возникает потребность в дополнительных линиях для передачи управляющих сигналов в частности сигнала стробирования.

Очень коротко поясним смысл термина «сигнал стробирования», или просто «строб». При передаче информации по параллельной магистрали всегда существует проблема, связанная с моментом ее считывания. Эта проблема является следствием некоторой электрической асимметрии выходных каскадов УСОМ передатчика и линий ОМ, вызванной технологическими причинами. Указанная асимметрия приво-

дит к разбросу времени установления сигналов на различных линиях ОМ. На синхронных магистралях эта проблема решается за счет введения некоторой задержки операции считывания относительно соответствующего синхроимпульса ОМ. На асинхронных магистралях момент считывания информации приемнику необходимо указывать специальным сигналом – стробом, который поступает из передатчика в приемник по отдельной линии с некоторой фиксированной задержкой относительно момента выставления данных на линиях ОМ.

- *Замкнутые магистрали*

Отличительной чертой магистралей этого типа является то, что между передатчиком и приемником существует обратная связь, суть которой состоит в следующем. Приемник, после считывания информации с линий ОМ обязан каким-либо сигналом (квитанцией) известить передатчик о завершении цикла обмена. Для передачи квитанции используют либо линии ШД, либо специально выделенные линии. При использовании корректирующих кодов квитанция может сообщить передатчику о возникшей ошибке. Существуют различные алгоритмы обмена по замкнутой магистрали, однако в любом случае передатчик не начинает новый цикл обмена до получения квитанции. При отсутствии квитанции в течение некоторого тайм-аута возникает прерывание, и управление передается операционной системе. Это позволяет предотвратить ошибки в системе, возникающие за счет сбоев в аппаратуре и внешних помех. Последнее особенно важно для аппаратуры промышленного применения, т.е. систем АСУ ТП.

Основным преимуществом замкнутых магистралей является повышенная надежность обмена по ОМ, что существенно повышает надежность вычислительной системы в целом. Основным недостатком замкнутых магистралей является то, что они требуют дополнительное оборудование для формирования и передачи квитанции. Кроме того, несколько увеличивается время цикла обмена из-за тайм-аута при ожидании квитанции.

- *Разомкнутые магистрали*

Отличительной чертой магистралей этого типа является то, что между передатчиком и приемником не существует никакой обратной связи. Передатчик, выставив на линии ОМ подлежащую передаче информацию, больше «не заботится» о том, считана она приемником или нет. Предполагается, что информация обязательно считана приемником и возможна инициализация нового цикла обмена.

Основным преимуществом разомкнутых магистралей является простота аппаратного и программного обеспечения ОМ, а следовательно, и меньшая стоимость. Кроме того, они имеют повышенную производительность при обмене. Основным недостатком разомкнутых магистралей является повышенная вероятность ошибок в системе, возникающих за счет сбоев в аппаратуре и внешних помех. Это существенно ограничивает область применений разомкнутых магистралей.

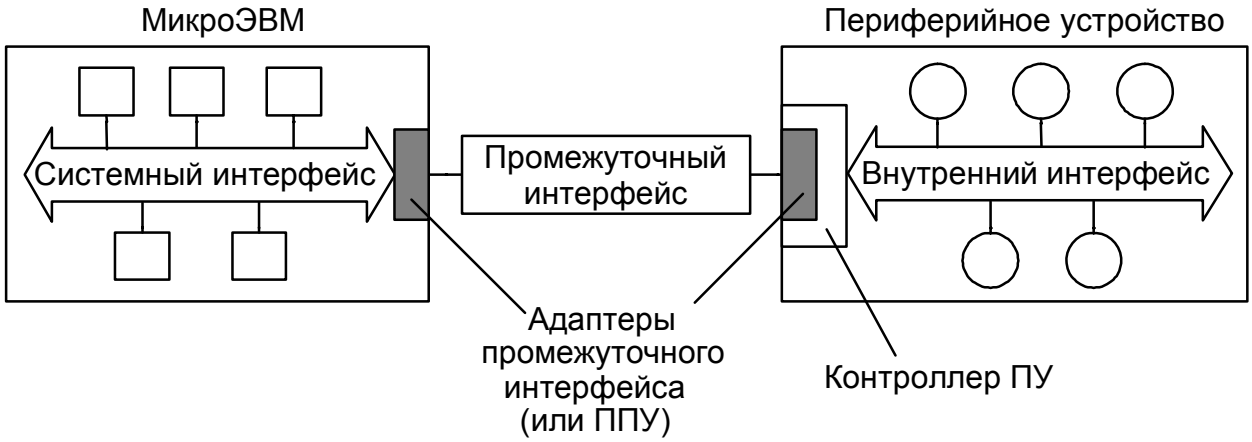
Принимая во внимание изложенное, следует отметить, что рассматриваемая ниже микроЭВМ на процессоре КР580ВМ80 (см. п. 7.4) построена на простейшем варианте синхронной разомкнутой магистрали. Синхронизация всех устройств и операций обмена по магистрали осуществляется от одного ГТИ.

7.2. ПРОМЕЖУТОЧНЫЙ ИНТЕРФЕЙС

К процессору микроЭВМ обычно подключается достаточно много ПУ. Это клавиатура, индикаторы, печатающие устройства, накопители, различные датчики и исполнительные устройства систем управления и т.д. ПУ свойственны различное быстродействие, различный набор управляющих сигналов, различные электрические параметры, т.е. их внутренний интерфейс (интерфейс ПУ), как правило, не совмес-

тим с системным интерфейсом микроЭВМ. Для сопряжения микроЭВМ с ПУ приходится использовать *промежуточный интерфейс*, поддерживаемый с обеих сторон специальными *адаптерами* (ППУ), т.е. между системным интерфейсом микроЭВМ и внутренним интерфейсом ПУ вводится промежуточный интерфейс последовательной или параллельной передачи данных (рис. 7.3). В принципе, каждое ПУ может иметь свой промежуточный интерфейс. Однако для упрощения и унифицирования аппаратуры сопряжения целесообразно использовать единый *стандартный промежуточный интерфейс*. Используя один и тот же стандартный промежуточный интерфейс последовательной или параллельной передачи данных к микроЭВМ, можно подключать различные ПУ.

Разнообразие ПУ и, прежде всего, их различное быстродействие (например, клавиатура и НМД) не позволяют в реальных микроЭВМ использовать только один тип промежуточного стандартного интерфейса. Между тем каждый из используемых в реальных микроЭВМ стандартных промежуточных интерфейсов позволяет подключать достаточно большие группы ПУ.



ППУ – приемопередающее устройство;
ПУ – переферийное устройство

Рис. 7.3. Включение промежуточного стандартного интерфейса

При дальнейшем изложении материала под стандартными промежуточными интерфейсами будут пониматься только два типа интерфейсов, которые можно назвать классическими – это последовательный интерфейс RS-232C и параллельный интерфейс CENTRONICS. Адаптеры указанных интерфейсов серийно выпускаются в виде БИС программируемых ППУ и могут использоваться с различными микропроцессорными комплектами. Так, в первом отечественном микропроцессорном комплекте КР580 (прототипом является микропроцессорный комплект фирмы Intel I8080) присутствуют два типа программируемых ППУ – КР580ВВ51 (прототип I8251) и КР580ВВ55 (прототип I8255), являющиеся соответственно адаптерами последовательного и параллельного интерфейсов. (Более подробно принципы последовательной и параллельной передачи данных, а также указанные ППУ рассматриваются в п. 8). Аналогичные ППУ присутствуют и в современных микропроцессорных комплектах. Например, ППУ UPD71051C и UPD71055 фирмы NEC или микросхемы серий 8211 и 8250/1645/16550 и их модификации фирмы Intel.

Наличие стандартного промежуточного интерфейса создает удобство в подключении различных ПУ к микроЭВМ. Однако этого недостаточно для ведения операций обмена с конкретным ПУ. Для каждого ПУ необходима специальная программа, которая, используя стандартный промежуточный интерфейс, будет вести обмен

с конкретным ПУ. Такие программы называют *драйверами устройств*. Обычно они входят в состав операционной системы.

Таким образом, в системах ввода/вывода современных микроЭВМ можно выделить, как минимум два уровня сопряжения ПУ с процессором и памятью. На первом уровне ППУ сопрягаются с процессором и памятью через системный интерфейс микроЭВМ, который комплексирует отдельные устройства микроЭВМ в единую вычислительную систему. На втором уровне сопряжения ППУ микроЭВМ посредством шин связи соединяются с аналогичными ППУ соответствующих ПУ, т.е. реализуется стандартный последовательный или параллельный промежуточный интерфейс.

Современные ПУ являются, как правило, сложными функциональными устройствами, для управления которыми одних драйверов устройств оказывается недостаточно. Драйверы могут оказаться недопустимо громоздкими и требовать больших вычислительных затрат со стороны центрального процессора. Кроме того, оперативность управления только через драйвер для быстродействующих ПУ может оказаться недостаточной. В связи с этим в состав сложных ПУ всегда входят специальные устройства управления, называемые *контроллеры ПУ.*

Следует отметить, что понятие "контроллер ПУ" охватывает очень широкий круг устройств. Контроллеры современных ПУ, таких как видеосистемы, жесткие диски и т.п., представляют собой сложнейшие устройства, имеющие собственные процессоры и память, т.е. являются специализированными микроЭВМ с соответствующим программным обеспечением. В то же время контроллером ПУ может быть и достаточно простое устройство, состоящее из нескольких логических схем. В ряде случаев контроллер ПУ может вообще отсутствовать. Тогда работой ПУ полностью управляет драйвер устройства, а сигналы, поступающие через промежуточный интерфейс, непосредственно воздействуют на узлы ПУ. Необходимо иметь в виду, что в функции контроллеров современных сложных ПУ входит не только организация операций обмена с ядром ЭВМ, но и управление функционированием ПУ в целом, например все действия, связанные с организацией системы физических записей на жестком диске.

При наличии контроллера, что характерно для большинства современных ПУ, управляющие сигналы и данные, переданные по промежуточному интерфейсу от микроЭВМ, первоначально поступают в контроллер ПУ и подвергаются соответствующей обработке. Это обстоятельство отражено на схеме включения промежуточного интерфейса (см. рис. 7.3), где адаптер промежуточного интерфейса является как бы частью контроллера ПУ. Передача данных и сигналов состояния от ПУ к микроЭВМ также происходит под управлением контроллера ПУ.

Подсистемы ввода/вывода больших ЭВМ имеют гораздо более сложную, многоуровневую иерархическую структуру интерфейсов, управляемую специализированными канальными процессорами. Подобные подсистемы обеспечивают обмен данными с множеством сложных устройств, таких как дисковые массивы, графические станции, серверы баз данных, другие ЭВМ, удаленные терминалы и т.п. Их описание выходит далеко за рамки настоящего раздела.

При дальнейшем изложении материала предполагается простейший случай двухуровневой системы сопряжения ПУ и ядра микроЭВМ с использованием одного стандартного промежуточного интерфейса.

7.3. МП С ФИКСИРОВАННОЙ СИСТЕМОЙ КОМАНД

В п. 3 уже рассматривались принципы функционирования элементарного гипотетического микропроцессора (термин "микропроцессор" и "процессор" далее используются как синонимы). Между тем для изучения принципов функционирования даже простейшей микроЭВМ необходимо выбрать конкретный тип процессора с конкретной системой команд и управляющих сигналов. Примером простейшего универ-

сального процессора (т.е. процессора с универсальной системой команд) аккумуляторного типа является I8080 (отечественный аналог КР580ВМ80А), выпущенный Intel в 1974 году. В этом же году на процессоре I8080 был спроектирован компьютер "Альтаир 8800", который некоторые эксперты называют первым персональным компьютером в истории развития техники. Именно это поколение 8-разрядных МП (I8080, I8085, Z80 и др.) стало широко применяться в управляющих микроЭВМ, контроллерах АСУ ТП, микрокомпьютерах общего назначения и в учрежденческой деятельности, в основном для обработки текстов.

Процессор I8080 имеет 8-разрядное АЛУ и УУ, выполненные на одном кристалле, содержащем около 5000 транзисторов. БИС МП имеет 40 выводов. Управляющее устройство выполнено на ПЛМ и недоступно пользователю, т.е. процессор имеет фиксированную систему команд. Напряжение питания ±5 В и +12 В. МП имеет двухфазную синхронизацию (F_1 и F_2) при тактовой частоте до 2,5 МГц и следующие шины:

- ША – 16-разрядная. Используется для адресации:
  - ОЗУ, ПЗУ (команды R/W) – 16 разрядов (адресное пространство составляет 64 К);
  - ПУ (команды I/O) – 8 разрядов (256 адресов);
- ШД – 8-разрядная, двунаправленная. Используется:
  - для приема операндов и команд от памяти и ПУ;
  - выдачи данных (результатов) в память и ПУ;
- ШУ – отдельно не оформлена и имеет 10 линий, по которым передаются 4 входных и 6 выходных сигналов.

Условное обозначение МП на схемах приведено на рис. 7.4.

Рассмотрим коротко, без подробных пояснений, назначение управляющих сигналов, учитывая, что ряд входных и выходных сигналов образуют как бы пары, отвечая за те или иные действия МП.

Управление прерыванием

INT – входной сигнал запроса прерываний от ПУ, воспринимаемый МП после выполнения текущей команды. Сигнал не воспринимается, если МП находится в режиме захвата или запрещения прерывания.

INTE – выходной сигнал разрешения прерывания. Этот сигнал отражает состояние внутреннего триггера "разрешение прерывания", который устанавливается только программно.

Управление режимом ожидания

READY – входной сигнал готовности, который сообщает о готовности устройства вести обмен с МП. При его отсутствии МП переходит в состояние ожидания. Позволяет синхронизировать работу МП и более медленной памяти или ПУ. Сигнал READY может задаваться как с ПУ, так и со вспомогательного таймера. В простейших устройствах этот вход не используется и подключается через сопротивление к источнику +5 В.

WAIT – выходной сигнал ожидания, подтверждающий, что МП находится в режиме ожидания.

Управление обменом информации

HOLD (HLD) – входной сигнал захвата шин от ПУ. Переводит буферы (см.с.78) ША и ШД МП в состояние Z , т.е. МП отключается от шин. Это позволяет ПУ занимать магистраль для инициализации обмена

HLDA – выходной сигнал подтверждения состояния захвата МП.

Управление чтением/записью

DBIN – выходной сигнал приема. Указывает памяти и ПУ, что ШД находится в режиме приема информации в МП, т.е. в режиме чтения.

$\overline{WR}$ – выходной сигнал выдачи. Используется для управления выдачей информации из МП в память и ПУ (режим записи). Активным является $\overline{WR}$ = 0.

Управление счетчиком команд

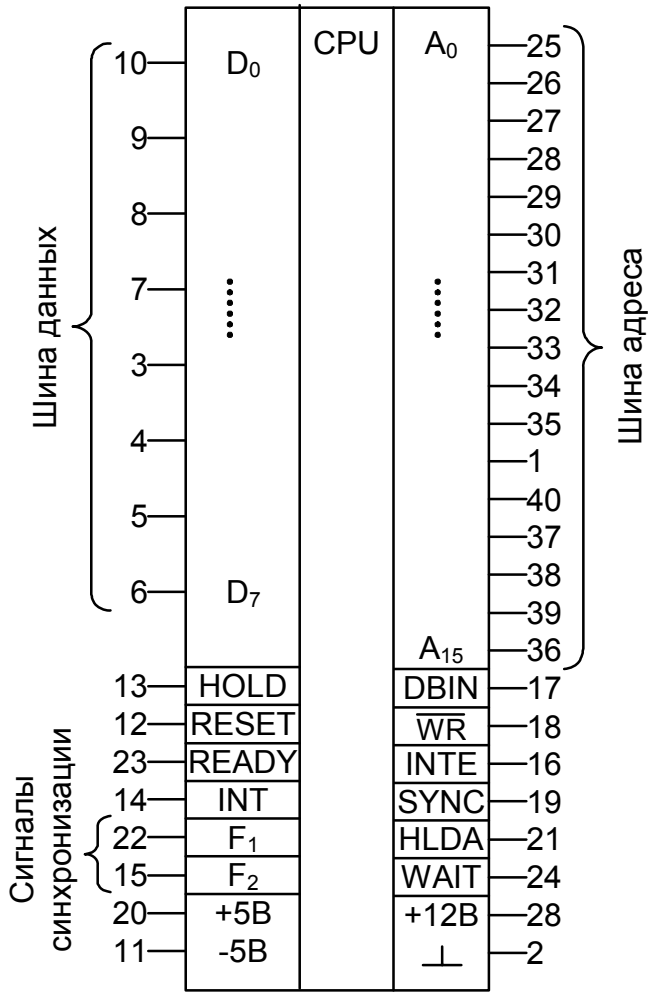


Рис. 7.4. Условное обозначение МП КР580ВМ80

RESET – входной сигнал сброса. Устанавливает в 0 счетчик адреса команд.

Синхронизация

SYNC – выходной сигнал синхронизации. Указывает на начало каждого нового машинного цикла.

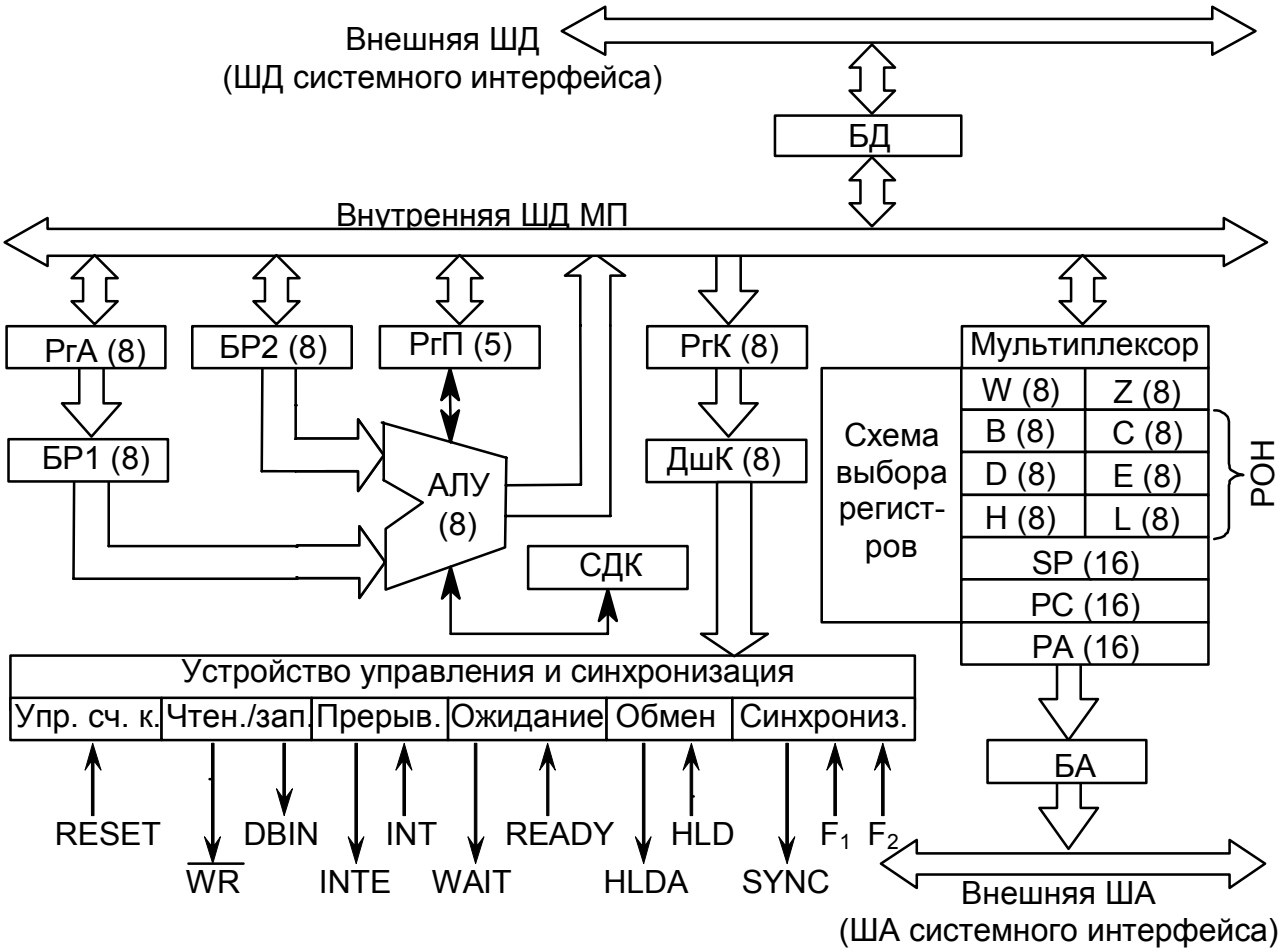
F_1, F_2 – синхропоследовательности, определяющие такт работы МП. Задаются кварцованным генератором.

Упрощенная функциональная схема МП изображена на рис. 7.5. На схеме изображены только функциональные связи между основными узлами МП. Цепи передачи управляющих сигналов, порождающих соответствующие микрооперации в узлах МП, на рисунке отсутствуют. В скобках указана разрядность устройств МП.

7.3.1. РЕГИСТРЫ ДАННЫХ

Для хранения участвующих в операции данных предусмотрены семь 8-разрядных регистров. РгА, называемый аккумулятором, предназначен для обмена информацией с памятью и ПУ, т.е. его содержимое может быть выдано на ШД либо число с ШД записано в него. При выполнении арифметических, логических операций

и операций сдвига он служит источником операнда. В него же всегда помещается



РгА – аккумулятор;
РгП – регистр признаков;
БР1, БР2 – буферные регистры;
РгК – регистр команд;
ДшК – дешифратор команд;
СДК – схема десятичной коррекции;
ША – шина адреса;
БД – буфер данных;
БА – буфер адреса;
SP – указатель стека;
PC – счетчик адреса команд;
PA – регистр адреса;
РОН – регистр общего назначения;
АЛУ – арифметико-логическое устройство

Рис. 7.5. Функциональная схема МП

результат выполнения операций.

Шесть других регистров называются B, C, D, E, H, L и образуют блок регистров общего назначения – РОН. Эти регистры программно доступны, и обращение к ним осуществляется посредством команд передачи данных. Причем обмен данными внутри МП (т.е. между РОН, АЛУ и аккумулятором) осуществляется по внутренней 8-разрядной шине через двунаправленный мультиплексор. РОН могут хранить как данные, так и адреса. Эти регистры можно использовать двояко – как одиночные восьмиразрядные регистры и как регистровые пары BC, DE, HL для хранения 16-разрядных двоичных чисел.

Регистры БР1, БР2, W, Z используются как буферные и программно недоступны (т.е. их содержимое посредством команд пользователь изменять не может).

Указатель стека SP служит для адресации стековой памяти и может хранить 16-разрядные адреса.

Счетчик адреса команд PC предназначен для хранения 16-разрядного адреса команды, а точнее, адреса текущего байта команды, поскольку команды могут зани-

мать 1, 2, 3 байта. После выборки из памяти текущего байта содержимое счетчика увеличивается на 1, т.е. формируется адрес следующего байта. При обращении к памяти (если используется косвенная адресация) в качестве адреса может использоваться содержимое любой регистровой пары РОН.

При выдаче адреса содержимое соответствующих регистров передается в 16-разрядный регистр адреса (РА), из которого далее через БА адрес поступает на ША системного интерфейса.

7.3.2. АРИФМЕТИКО-ЛОГИЧЕСКОЕ УСТРОЙСТВО

МП имеет 8-разрядное АЛУ, которое позволяет выполнять 4 арифметические операции (сложение с передачей переноса в младший разряд и без него, вычитание с передачей займа в младший разряд и без него), четыре вида логических операций (конъюнкция дизъюнкция, неравнозначность, сравнение), а также 4 вида циклических сдвигов.

При выполнении арифметических операций одним из операндов всегда является содержимое аккумулятора. Результат всегда помещается в аккумулятор. Циклический сдвиг выполняется только над содержимым аккумулятора.

Предусмотрена возможность выполнения арифметических операций над десятичными числами. При этом в байт укладываются две десятичные цифры в коде 8421. При рассмотрении операций десятичной арифметики отмечалось, что может потребоваться коррекция результата, т.е. прибавление к нему числа $0110_{(2)}$. Такая коррекция в каждой тетраде результата осуществляется схемой десятичной коррекции (СДК).

7.3.3. РЕГИСТР ПРИЗНАКОВ

Ранее отмечалось, что РгП называют еще регистром флажков и обозначают часто буквами Ф или F. Это 8-разрядный регистр, в котором используются только 5 разрядов. Он предназначен для хранения ряда признаков, выявляемых в числе, которое является результатом операции в АЛУ или РОН. Триггеры регистра имеют следующее назначение:

T_c (триггер переноса) – устанавливается в 1 при наличии переноса из старшего разряда при выполнении арифметических операций (c=1 – перенос есть; c=0 – переноса нет). Кроме того, запоминает содержимое выдвигаемого из аккумулятора разряда при выполнении операции сдвига.

T_z (триггер нуля) – устанавливается в состояние логической 1, если результат операции в АЛУ или операции приращения содержимого РОН равен 0 (z=1 – результат равен 0, z=0 – результат не 0).

T_s (триггер знака) – устанавливается в состояние, соответствующее значению старшего разряда результата операции в АЛУ или операции приращения содержимого РОН (s=0 – результат положительный, s=1 – результат отрицательный).

T_p (триггер четности) – устанавливается в состояние логической 1, если число единиц в разрядах результата четно (p=1 – вес результата четный, p=0 – нечетный).

T_v (триггер дополнительного переноса) хранит перенос из 3-го разряда, возникающий при выполнении операции в АЛУ.

7.3.4. БЛОК УПРАВЛЕНИЯ

Состоит из регистра команд, куда принимается первый байт команды, дешифратора команд и непосредственно управляющего устройства, формирующего управляющие сигналы, под действием которых выполняются последовательности микроопераций в отдельных узлах МП. Как уже отмечалось, управляющее устройство выполнено на ПЛМ, т.е. микропрограммы хранятся за счет системы жестких связей и не могут быть изменены пользователем. Четыре входных и шесть выходных сигналов управляющего устройства при наличии системного контроллера (см. п. 7.4) позво-

ляют процессору управлять вычислительными системами достаточно сложной структуры.

7.3.5. БУФЕРЫ

Буферы адреса и данных связывают МП с внешними шинами адреса и данных (шинами системного интерфейса). В качестве выходных каскадов в буферах используются логические элементы с тремя состояниями. Это позволяет процессору отключаться от внешних шин и предоставлять их в распоряжение ПУ. Буфер ШД двунаправленный, что позволяет использовать ШД в полудуплексном режиме для приема и передачи информации в неперекрывающиеся интервалы времени.

Принцип двунаправленного обмена данными между внутренней и внешней ШД можно пояснить схемой, изображенной на рис. 7.6.

Следует помнить, что передача информации по шинам как внутренним, так и внешним осуществляется в параллельном коде, т.е. выходные и входные логические элементы буфера ШД имеют 8- канальную структуру.

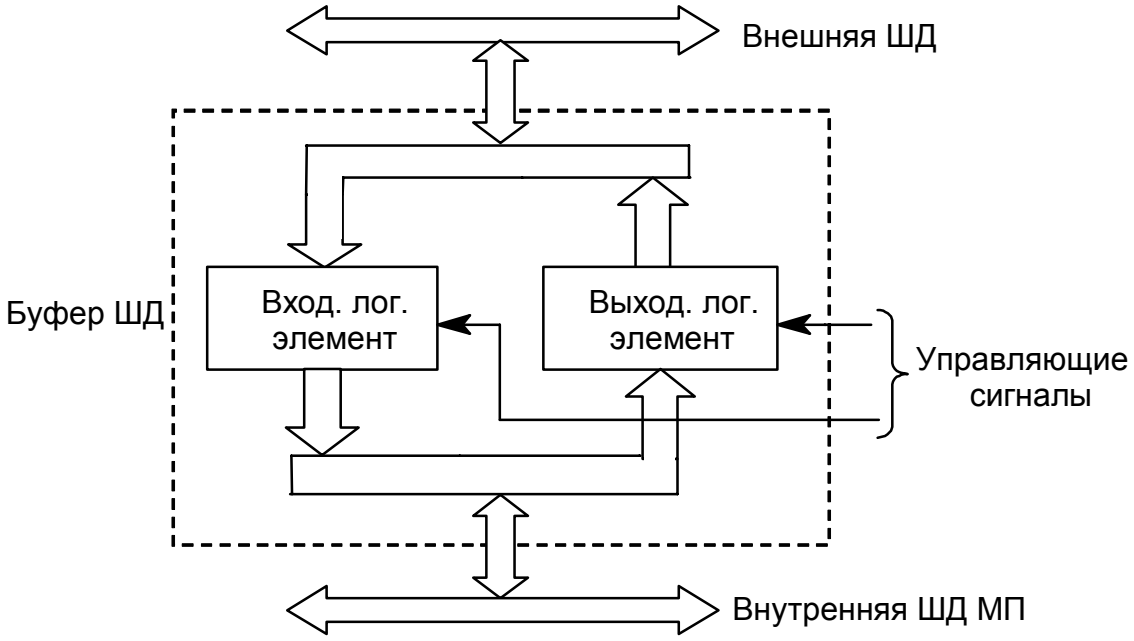


Рис. 7.6. Функциональная схема буфера ШД МП

7.3.6. МП С ТОЧКИ ЗРЕНИЯ ПРОГРАММИСТА

С точки зрения пользователя, реализация физических процессов, протекающих в микросхеме, не представляет особого интереса, как и физическая реализация отдельных узлов МП. В распоряжение пользователя предоставляется ряд формальных устройств МП, с которыми он может производить те или иные манипуляции посредством команд программы. Для общения с тем или иным устройством пользователю необходимо знать адрес или номер этого устройства.

Структура МП I8080 (КР580), с точки зрения программиста, представлена на рис. 7.7. Аккумулятор и регистр признаков (флагов) являются половинами одного 16-битного регистра, но пользователь может обращаться к ним как к отдельным 8-битовым регистрам. В документации на МП КР580 содержимое РгП названо *словом состояния процессора* (PSW). Однако некоторые авторы под PSW понимают не только содержимое РгП, но и содержимое аккумулятора, т.е. PSW ≡ (A)(F). Такая трактовка PSW (регистра PSW) используется при дальнейшем изложении материала.

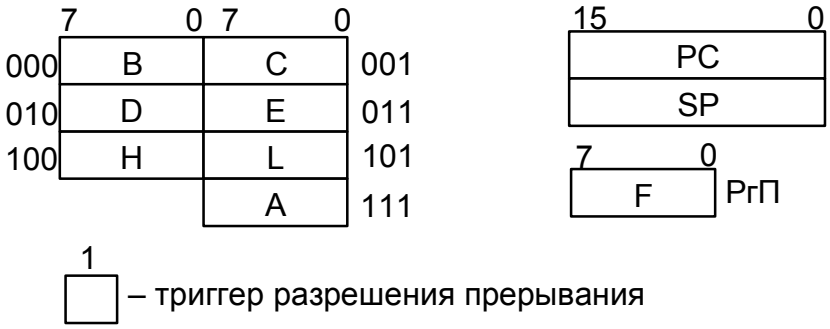
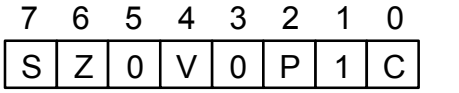


Рис. 7.7. МП с точки зрения программиста

Таким образом, пользователь имеет в своем распоряжении:
- восьмибитовые регистры A, B, C, D, E, H, L, каждый из которых имеет свой трехразрядный адрес (номер);
- 16-битовые регистровые пары BC, DE, HL, определяемые по имени старшего регистра пары (т.е. B, D, H);
- 16-битовые регистры PC, SP, PSW;
- 8-битовый РгП(F), который можно рассматривать отдельно от PSW;
- однобитовый регистр (триггер) разрешения прерывания.

Более подробно РгП (F) изображен на рис. 7.8.



s – знаковый (0 – положительный результат, 1 – отрицательный результат);
z – нулевой результат (1 – результат равен нулю, 0 – результат не равен нулю);
v – дополнительный перенос из 3-го в 4-й разряд (1 – есть перенос, 0 – нет переноса);
p – четность числа единиц в результате; (1 – четное число, 0 – нечетное число);
c – перенос из 7-го разряда (1 – есть перенос, 0 – нет переноса)

Рис. 7.8. Регистр признаков

7.4. МП-УСТРОЙСТВО НА ОСНОВЕ МП КР580ВМ80А

Упрощенная структурная схема вычислительного устройства на базе МП I8080 (КР580ВМ80А) представлена на рис. 7.9. Это простейшая микроЭВМ минимальной конфигурации, структура которой является частным случаем обобщенной (см. рис. 7.1).

Представленная схема включает все основные функциональные блоки, за исключением источника питания. ПЗУ может быть использовано для хранения программы, а ОП для хранения данных, поступающих от ПУ (через ППУ), а также результатов работы программы. Предполагается, что ОП и ПЗУ охвачены единым полем адресов.

К шинам адреса и данных системной магистрали, даже в простейшей микроЭВМ, подключено достаточно много устройств: ОП, ПЗУ, несколько ППУ. Однако нагрузочная способность выходов МП КР580, в силу технологических особенностей, весьма мала. К любому выходу МП допускается подключать не более одного входа микросхемы ТТЛ, поэтому в шины адреса и данных включаются специальные буферы, причем ШД требует двунаправленного буфера. Для построения таких буферов предусмотрены микросхемы *шинных формирователей* КР580ВА86 и КР580ВА87.

Общие принципы функционирования МП устройства следующие. Из МП на ША (16 разрядов) выдается адрес очередной команды. В этот момент МП еще «не знает», сколько байт занимает данная команда. Первый байт команды, выбранный из памяти (в частном случае из ПЗУ), пересылается по внутренней ШД в РгК. Выход РгК связан с дешифратором команд, который определяет тип выполняемой операции. При этом к содержимому СчК добавляется 1, т.е. формируется адрес следующего байта, а УУ вырабатывает ряд сигналов, позволяющих выполнить те или иные микрооперации. После этого возможны два варианта дальнейших действий:

- Если команда однобайтовая, то она выполняется, а содержимое счетчика адреса (PC) = (PC) + 1 является адресом следующей команды.
- Если команда содержит более одного байта (2 или 3) и для ее выполнения требуется вызов дополнительных байтов, то содержимое счетчика адреса команд (PC) = (PC) + 1 является адресом следующего байта той же команды.

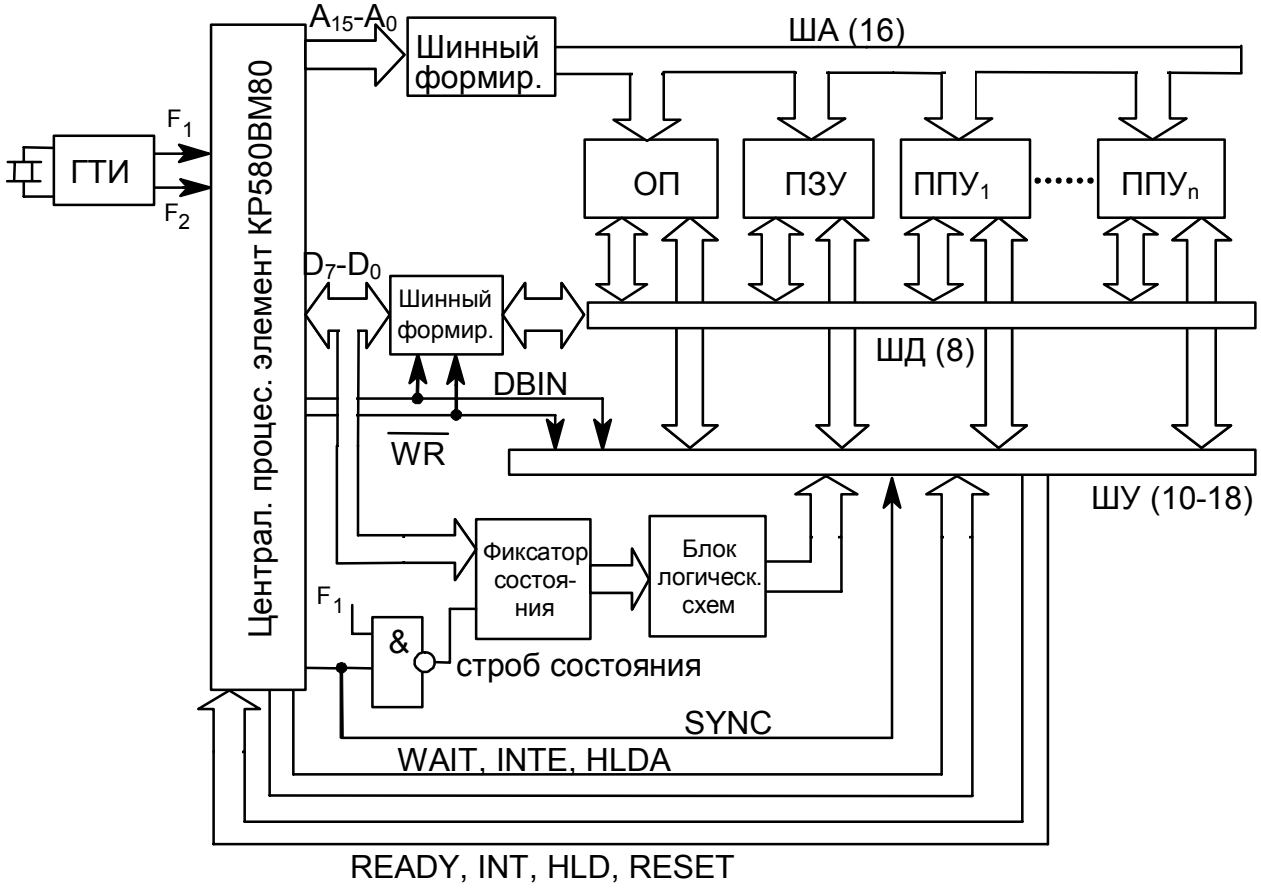


Рис. 7.9. Упрощенная структурная схема микроЭВМ на базе МП КР580ВМ80

Рассмотрим более подробно процесс выполнения команды. Этот процесс разбивается на *машинные циклы*, которые обозначаются М1...М5. Число циклов в одной команде может быть от одного до пяти. В свою очередь, каждый машинный цикл состоит из тактов, обозначаемых Т1...Т5. В одном машинном цикле может быть от трех до пяти тактов. Имеется в виду 5 типов тактов, поскольку в каждом такте выполняется определенное действие по реализации машинного цикла. При этом количество тактов как временных интервалов может быть значительно больше за счет тактов Т2, о чем речь пойдет ниже. В каждом машинном цикле производится одно обращение к памяти или ППУ в разных вариантах. Каждый такой вариант обращения называется *состоянием* цикла. Всего в МП КР580 возможно 10 состояний машинного цикла. Это выборка первого байта команды, чтение из памяти, запись в память, чте-

ние из стека, запись в стек, ввод из ППУ, вывод через ППУ, подтверждение прерывания, подтверждение останова, подтверждение прерывания при останове. При этом первым машинным циклом любой команды всегда является выборка первого байта команды.

Во всех машинных циклах первые три такта (T1,T2,T3) используются для организации обмена с памятью и ППУ. Такты T4 и T5 (если они есть) – для выполнения внутренних операций в МП. Таким образом, процесс выполнения команд состоит из стольких машинных циклов, сколько обращений к памяти или ППУ требуется для ее исполнения.

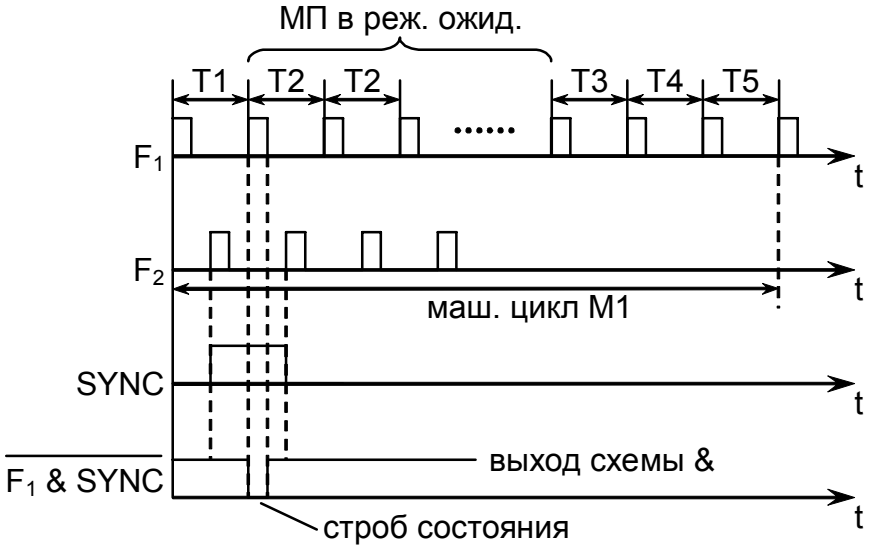


Рис. 7.10. Временная диаграмма цикла M1

На рис. 7.10 представлена временная диаграмма цикла M1 из пяти тактов (первый машинный цикл любой команды). Отсчет тактов производится от положительных фронтов импульса F1. Действия процессора по реализации машинного цикла M1 состоят в следующем:

T1 – содержимое PC выдается на ША, адрес принимается памятью, где начинается чтение байта команды из ячейки.

T2 – проверяется наличие сигнала на входе READY (уровень логической 1). Этот сигнал подается на вход МП через интервал времени, достаточный для завершения процесса чтения из памяти. Если на входе READY сигнал отсутствует (действует логический 0), то МП устанавливается в режим *ожидания*, в котором каждый следующий такт рассматривается как T2 до тех пор, пока не появится сигнал READY. С приходом этого сигнала МП выходит из режима ожидания и переходит в такт T3.

T3 – байт с ШД принимается в МП и помещается в регистр команд (РгК).

T4 – происходит анализ принятого байта и выяснение потребности в дополнительном обращении к памяти. Если дополнительных обращений не требуется (команда однобайтовая и операнды находятся в регистрах процессора), то в этом же такте или с использованием дополнительного такта T5 выполняются предусмотренные командой микрооперации.

T5 – дополнительный такт.

Если требуется дополнительное обращение к памяти, то после T4 цикл M1 завершается и происходит переход к циклу M2.

Забегая вперед, отметим, что КОП всегда находится в первом байте команды. Если команда двух- или трехбайтовая, то в остальных байтах находятся данные или

адрес. Содержимое этих байтов помещается в аккумулятор или буферные регистры. Так, например, в команде MOV (запись аккумулятора в ячейку памяти) двухбайтовый адрес, который следует за КОП, помещается в регистровую пару WZ, а затем, при исполнении, он передается через мультиплексор непосредственно в РА и далее через буфер на ША.

В каждом машинном цикле в такте Т1 по переднему фронту F2 МП выдает сигнал синхронизации SYNC, т.е. на выходе SYNC появляется уровень логической 1. Одновременно с этим сигналом в такте Т1 МП выставляет на ШД 8-разрядное *управляющее слово*, которое несет в себе полную информацию о микрооперациях в текущем машинном цикле. Так, например, 1 в разряде D0 управляющего слова является сигналом подтверждения прерывания INTA. Наличие 1 в разряде D2 означает, что в данном машинном цикле на ША установлено содержимое указателя стека (регистр SP). Наличие 1 в разряде D3 означает, что МП в состоянии останова. В момент прихода импульса F1, означающего начало такта Т2, на схеме "&" (см. рис. 7.9) вырабатывается импульс, называемый *строб состояния*. Этот строб разрешает запись управляющего слова с ШД во внешний регистр, названный на схеме *фиксатор состояния*.

Используя это слово или его часть, специальные логические схемы вырабатывают системные управляющие сигналы для обращения к памяти и ППУ. В общем случае фиксатор состояния и блок логических схем называются *системным* контроллером. Эти, а также некоторые другие вспомогательные схемы, в частности шинный формирователь, оформлены в виде специальной БИС КР580ВК28. Однако в простейших микроЭВМ часто требуются только 4 управляющих сигнала – R, W, IN, OUT. В связи с этим необходимость в БИС ВК28 отпадает, а используют какой-либо управляемый регистр и 2-3 логические схемы.

7.5. ФОРМАТЫ ДАННЫХ МП КР580

Основной формат данных изображен на рис. 7.11.

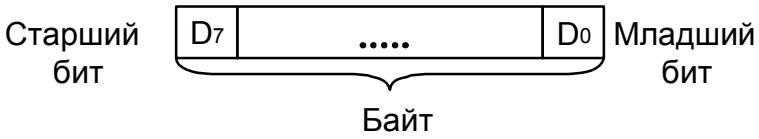


Рис. 7.11. Основной формат данных

В микроЭВМ байт данных может интерпретироваться следующим образом:

- Целое число без знака

$255_{(10)} = 1111\ 1111_{(2)} = FF_{(16)}$ (FFH)

. . . . . .

$0_{(10)} = 0000\ 0000_{(2)} = 00_{(16)}$ (00H).

- Число со знаком в дополнительном коде

$+127_{(10)} = 0111\ 1111_{(2)} = 7F_{(16)}$

. . . . . .

$0_{(10)} = 0000\ 0000_{(2)} = 00_{(16)}$

$-1_{(10)} = 1111\ 1111_{(2)} = FF_{(16)}$

. . . . . .

$-128_{(10)} = 1000\ 0000_{(2)} = 80_{(16)}$.

- Двухразрядное двоично-десятичное число

$99_{(10)} = 1001\ 1001_{(2)} = 99_{(16)}$

. . . . . .

$0_{(10)} = 0000\ 0000_{(2)} = 00_{(16)}$.

- Логический байт данных.

В ряде случаев используется двухбайтовый формат данных:

- Двухбайтовое число со знаком в дополнительном коде

$+32767_{(10)} = \underbrace{0111\ 1111}_{\text{1-й байт}}\ \underbrace{1111\ 1111}_{\text{2-й байт}}{}_{(2)} = 7F\ FF_{(16)}$

. . . . . .

$-32768_{(10)} = 1000\ 0000\ \ 0000\ 0000_{(2)} = 80\ 00_{(16)}.$

- Двухбайтовое число без знака

$65535_{(10)} = 1111\ 1111\ \ 1111\ 1111_{(2)} = FF\ FF_{(16)}$

. . . . . .

$0_{(10)} = 0000\ 0000\ \ 0000\ 0000_{(2)} = 00\ 00_{(16)}.$

7.6.ФОРМАТЫ КОМАНД МП 580ВМ80

Для команд используются одно-, двух-, трехбайтовые форматы, причем код операции (КОП) занимает всегда 1 байт. Кроме того, следует помнить, что ША имеет 16 разрядов, т.е. позволяет адресоваться к памяти в 64К однобайтовых ячеек. Следовательно, в этом случае для адреса требуется 2 байта.

В то же время, используя команды INPUT/OUTPUT, программист может адресоваться к 256- регистрам ППУ. Следовательно, в этом случае для адреса требуется только 1 байт, который передается по младшим разрядам ША $A_7...A_0$. (Этот же байт дублируется в старших разрядах ША $A_{15}...A_8$, т.е. если адрес регистра F8, то на ША присутствует F8F8).

В общем случае форматы команд МП имеют вид, показанный на рис. 7.12. В первом байте помещается КОП, а во втором однобайтовый операнд или номер реги-

КОП – код операции

Рис. 7.12. Форматы команд МП 580ВМ80

стра ППУ (рис. 7.12, а, б).

В трехбайтовом формате (рис. 7.12, в) первый байт содержит код операции (КОП). Два следующих байта содержат соответственно младший и старший байты адреса ячеек ОП (ПЗУ) либо двухбайтовый операнд.

7.7. СПОСОБЫ АДРЕСАЦИИ

Способы адресации рассмотрим очень коротко, поскольку все типы адресации в общем виде разобраны ранее.

Прямая адресация

В этом случае источником или приемником операнда являются ячейки памяти или регистр ППУ.

Адрес ячейки памяти или регистра ППУ записывается в команде:

- двухбайтовая команда – адрес регистра ППУ;
- трехбайтовая команда – адрес ячейки памяти.

Регистровая адресация

В этом случае источником или приемником операнда является РОН. Номер РОН записывается в коде операции. Команды *однобайтовые* (номер РОН – 3 разряда).

Косвенная адресация

В этом случае источником или приемником операнда является ячейка памяти. Команды *однобайтовые*. Адрес ячейки памяти находится в регистровой паре, например HL. Есть команды инкремента и декремента (± 1) содержимого регистровых пар, что позволяет модифицировать адрес.

Стековая адресация

В этом случае источником или приемником операнда является ячейка памяти. Команды *однобайтовые*.

Адрес вершины стека находится в указателе стека SP. При заполнении стек растет в сторону уменьшения адресов, т.е. формируется стек, который ранее был назван перевернутым. Начальный адрес стека устанавливается в указателе стека (SP) программным путем. Уже отмечалось, что SP имеет 16 разрядов, поэтому, если начальным адресом стека выбрать последний адрес всего адресного пространства, то стек может иметь емкость 64 Кбайт.

Обмен со стеком (т.е. загрузка/выгрузка) производится двухбайтовыми словами. При этом загрузка в стек сводится к следующей последовательности действий со стороны МП:

- выполняется операция (SP) = (SP) - 1;
- по новому адресу записывается старший байт вводимого двухбайтового слова;
- выполняется операция (SP) = (SP) - 1;
- по новому адресу записывается младший байт (второй) вводимого двухбайтового слова.

Выгрузка стека сводится к следующему:

- считывается младший байт по адресу (SP);
- выполняется операция (SP) = (SP) + 1;
- считывается старший байт по новому адресу;
- выполняется операция (SP) = (SP) + 1.

Непосредственная адресация

В этом случае операнд находится в самой команде. Операнд может быть одно- и двухбайтовым. В соответствии с этим команды могут быть *двухбайтовыми* и *трехбайтовыми* (первый байт всегда занимает КОП).

Следует помнить, что в трехбайтовой команде младшие разряды 16-битового числа содержатся во втором байте команды, а старшие в третьем.

7.8. СИСТЕМА КОМАНД МП 580

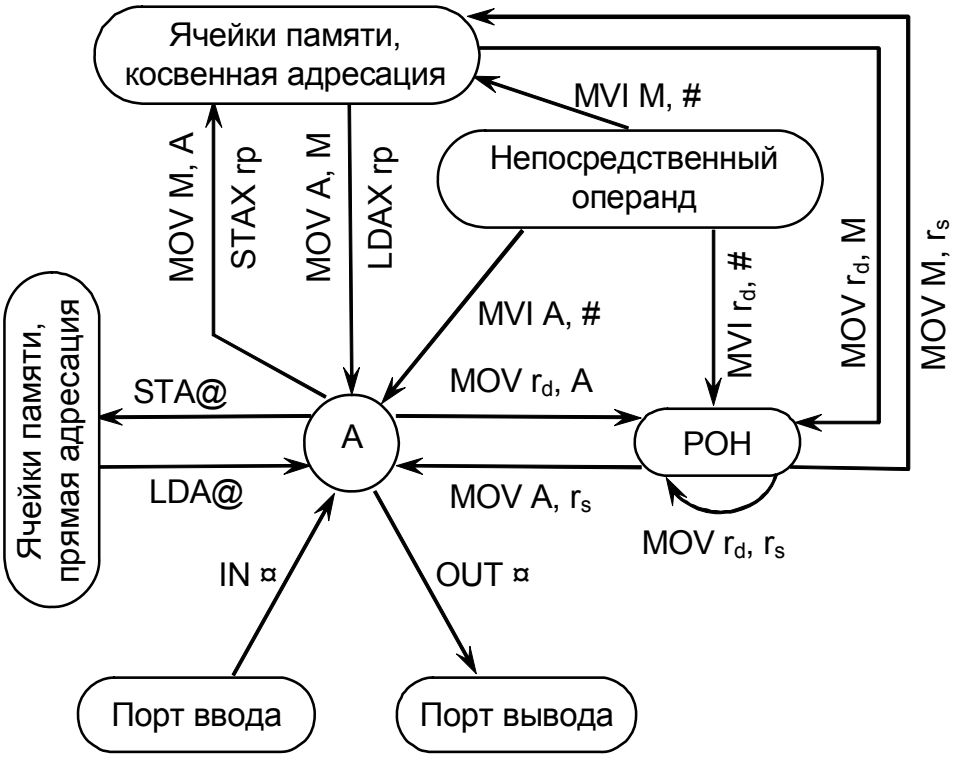
Для программирования микроЭВМ на базе МП комплекта КР580 используется 244 команды. Ниже очень коротко будет рассмотрена только часть команд, необходимая для программирования простых задач. Таблицы команд и краткие пояснения к ним можно найти во многих источниках. Наиболее подробные комментарии представлены в журналах "Радио" за 1982 год.

Следует помнить, что любая машинная команда – это двоичное кодовое слово определенной длины. Однако пользоваться двоичными кодами при программировании человеку крайне трудно, поэтому для программирования используют язык *ассемблера* или просто *ассемблер* (см. п. 1.4). В этом языке вместо кодовых комбинаций используется мнемоническая форма записи операций, выполняемых в БИС. Каждой команде на языке ассемблера соответствует одна команда на языке кодовых комбинаций. Ниже будут рассмотрены команды именно на языке ассемблера.

Перед исполнением программа должна быть переведена на язык кодовых комбинаций и помещена в память микроЭВМ. Такая трансляция производится автоматически специальной программой, называемой ассемблером. Если ассемблер отсутствует, то перед вводом программы такой перевод делают вручную. В этом случае ввод команд в микроЭВМ осуществляется обычно в шестнадцатеричной системе счисления.

Все команды МП КР580 целесообразно разделить на группы, например, следующим образом:

- Пересылки однобайтовые для обмена операндами и результатами между аккумулятором, РОН, памятью, регистрами ППУ.
- Пересылки двухбайтовые для обмена адресами, операндами, результатами между SP, регистровыми парами, парами ячеек памяти, стеком.
- Операции в аккумуляторе, которые делятся на арифметические, логические (в аккумуляторе и регистре флагов), сдвиги.
- Арифметические операции в РОН и памяти.
- Команды управления.



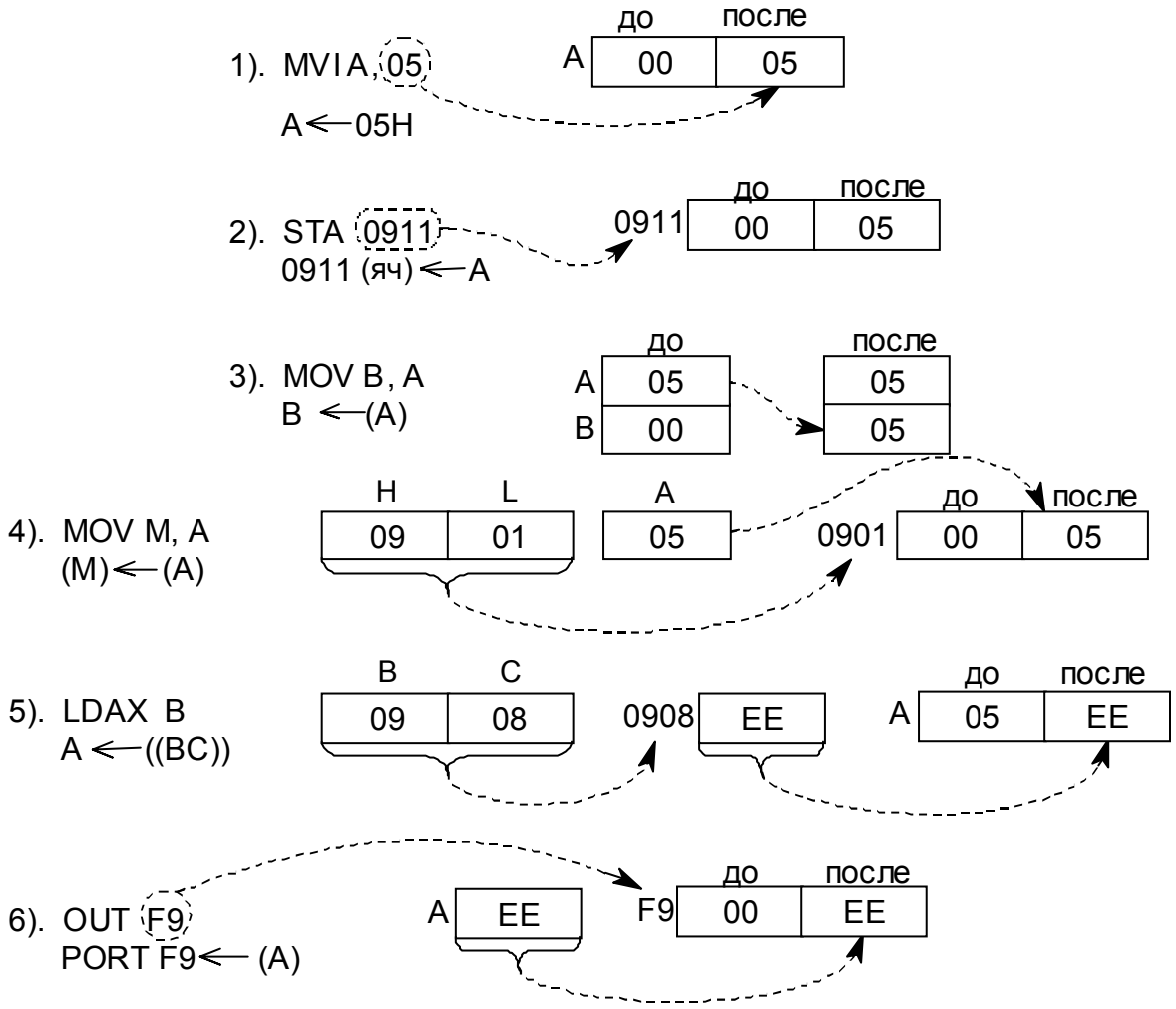
A – аккумулятор; rp – регистровая пара (rp = B|D (BC или DE));
r_s – регистр-источник; r_d – регистр-приемник (r_s, r_d = B| C| D| E| H| L);
– однобайтовый операнд; @ – двухбайтовый адрес;
¤ – однобайтовый номер (адрес) регистра ППУ; M – ячейка памяти с адресом, хранимым в HL (в команде с косвенной адресацией)

Рис. 7.13. Схема взаимодействия узлов МП при выполнении однобайтовых пересылок

7.8.1. ПЕРЕСЫЛКИ ОДНОБАЙТОВЫЕ

Команды этой группы *не изменяют* содержимого регистра признаков РгП (F). Принципы их выполнения иллюстрирует рис. 7.13.

Ниже рассмотрим в качестве примера последовательность из нескольких команд однобайтовых пересылок. Операнды и адреса записаны в шестнадцатеричной системе счисления. Кроме того, здесь и далее предполагается, что аккумулятор, ячейки памяти и регистры первоначально (до записи) содержат 00:



7.8.2. ПЕРЕСЫЛКИ ДВУХБАЙТОВЫЕ

Команды этой группы *не изменяют* содержимого РгП (F). Принципы их выполнения иллюстрирует рис. 7.14. Предполагается, что PSW≡(A)(F).

Поясним две команды:

XCHG – это обмен содержимым пар HL и DE, причем (H) ↔ (D), (L) ↔ (E);

XTHL – это обмен содержимым пары HL и верхушки стека. Значение SP не изменяется, при этом (L) ↔ ((SP)) и (H) ↔ ((SP)+1).

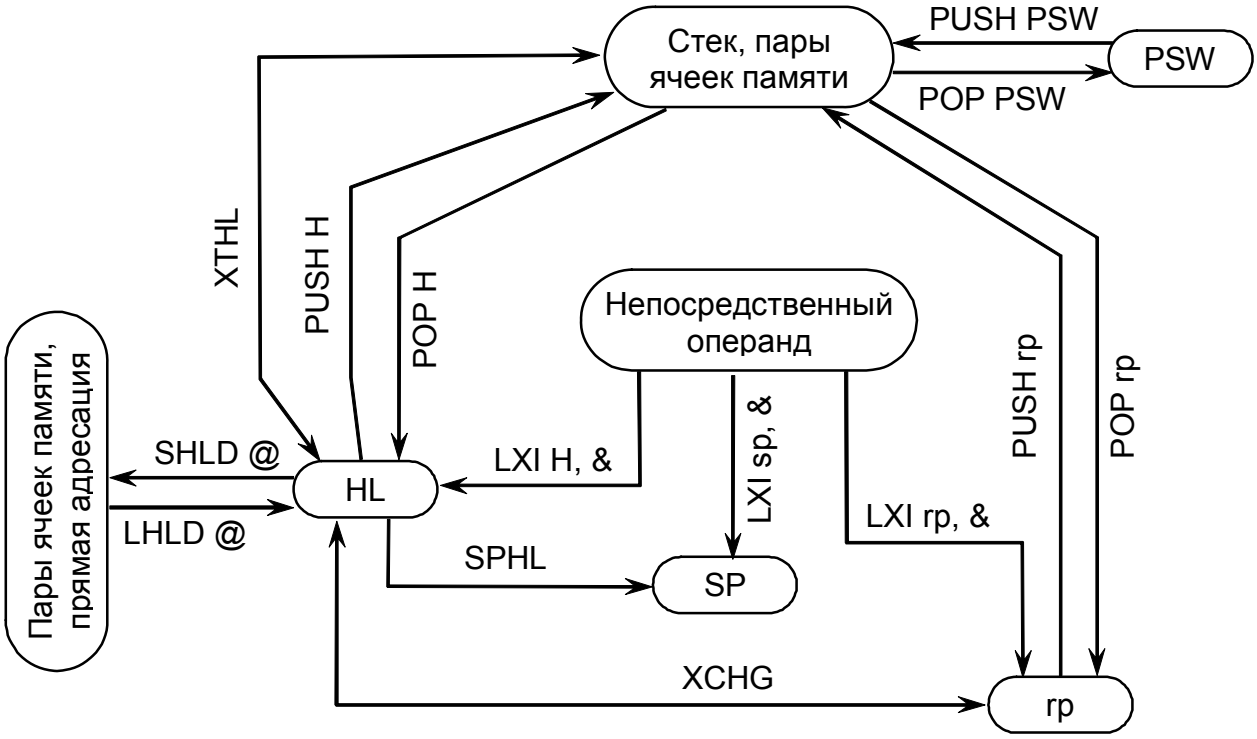
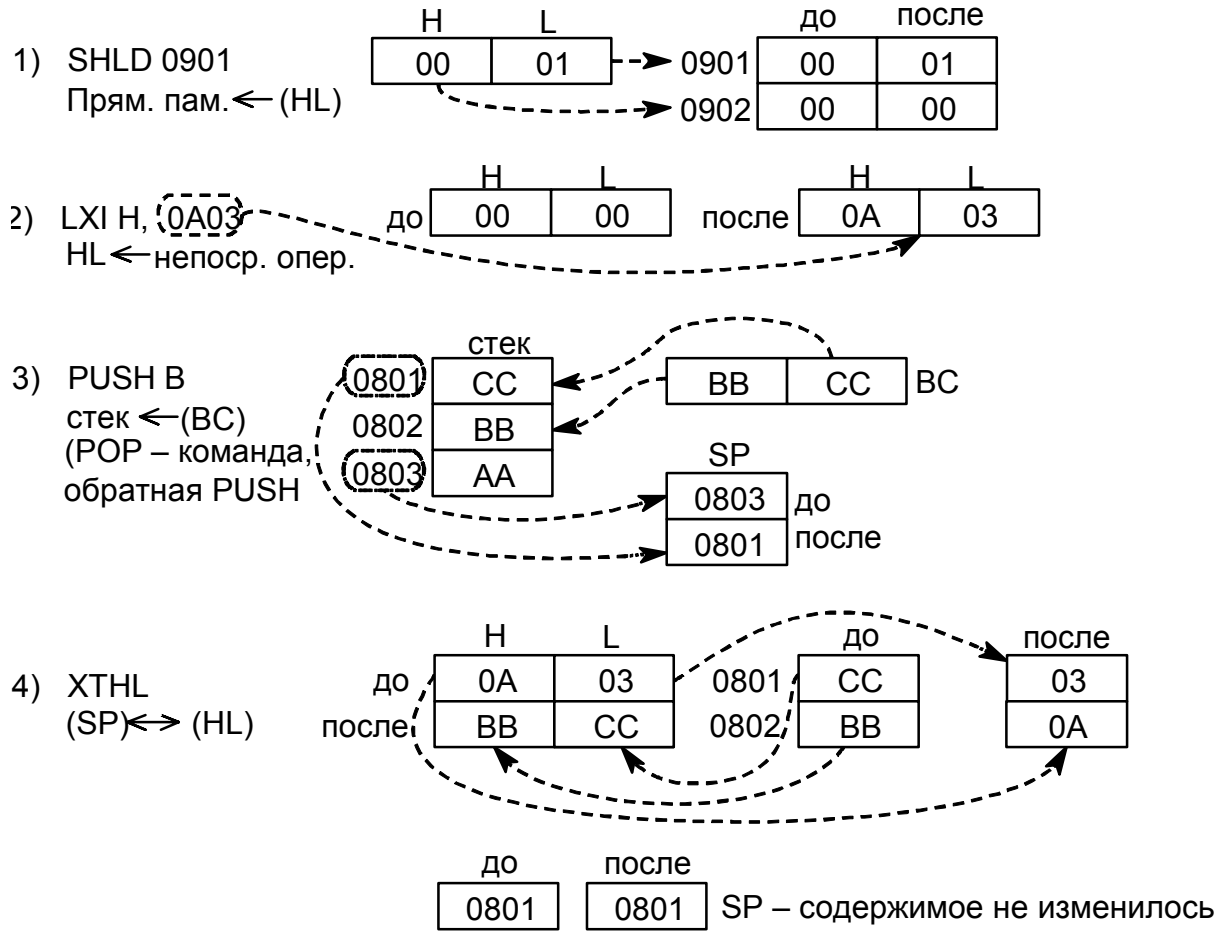


Рис. 7.14. Схема взаимодействия узлов МП при выполнении двухбайтовых пересылок: & – двухбайтовый операнд; остальные обозначения аналогичны рис. 7.13.

Рассмотрим в качестве примера несколько команд двухбайтовых пересылок.

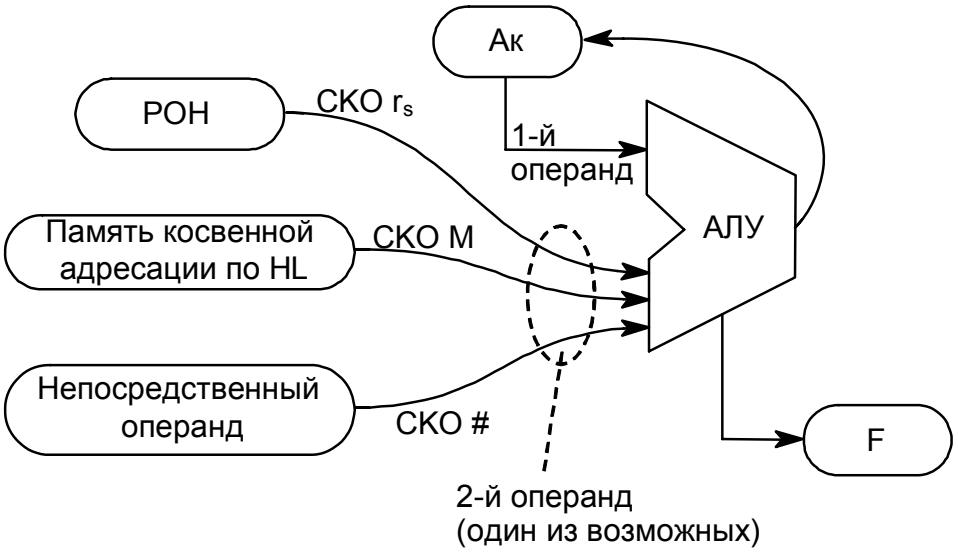


7.8.3. ОПЕРАЦИИ В АККУМУЛЯТОРЕ

Команды этой группы *изменяют* содержимое РгП (F) в соответствии с содержимым аккумулятора (рис. 7.15).

Выполняемые операции подразделяются на *двуместные и одноместные*.

Операции двуместные



РОН – регистр общего назначения;
АЛУ – арифметико - логическое устройство;
СКО – символьный код операции;
АК - аккумулятор

Рис. 7.15. Схема взаимодействия узлов МП при выполнении операций в аккумуляторе

В общем случае РОН = A| B| C| D| E| H| L; СКО – символьный код операции. Остальные обозначения аналогичны приведённым на рис. 7.13.

В свою очередь, двуместные операции в аккумуляторе делятся на арифметические и логические.

- *Арифметические операции*:
  - сложение содержимого аккумулятора с содержимым РОН, ячеек памяти (косвенная адресация по HL), непосредственным операндом;
  - сложение содержимого аккумулятора и тех же операндов и бита переноса С (часто обозначают CY);
  - вычитание из содержимого аккумулятора содержимого РОН, ячеек памяти (косвенно адресуемых по HL) или непосредственного операнда;
  - вычитание из содержимого аккумулятора тех же операндов и бита переноса;
  - сравнение содержимого аккумулятора с содержимым РОН, ячейками памяти (косвенно адресуемых по HL), непосредственным операндом.

В последнем случае вычисляется разность (А) – (операнд), которая никуда не записывается (т.е. А не изменится), а используется для установки флагов в регистре признаков F.

Рассмотрим несколько примеров:

ADD B A ← (A) + (B);
SUB M A ← (A) - (ячейка (M));
ACI 07 A ← (A) + 07 + C;

CMP D (A) - (D) → установка F, (A) – не изменилось.

- *Логические (побитовые) операции*:

  - конъюнкция содержимого аккумулятора с содержимым РОН, ячейки памяти (косвенно адресуемой по HL), непосредственным операндом.
  - дизъюнкция содержимого аккумулятора с содержимым РОН, ячейки памяти (косвенно адресуемой по HL), непосредственным операндом.
  - сложение по модулю 2 с содержимым РОН, ячейки памяти (косвенно адресуемой по HL), с непосредственным операндом.

Рассмотрим несколько примеров:

ANA D A ← (A) & (D);
ANI A4 A ← (A) & A4;
ORA M A ← (A) ∨ (ячейка (M));
XRA E A ← (A) ⊕ (E);
XRI F4 A ← (A) ⊕ F4.

Операции одноместные

- *Арифметическая операция*:

DAA – десятичная коррекция аккумулятора при работе с двоично-десятичными числами.

- *Логические операции*:

CMA – инверсия аккумулятора;
STC – установка бита C (т.е. C ← 1);
CMC – инверсия бита C (т.е. C ← $\overline{C}$).
Две последние команды выполняются в РгП (F).

- *Сдвиги на 1 разряд*:

Примеры реализации сдвиговых команд приведены на рис. 7.16.

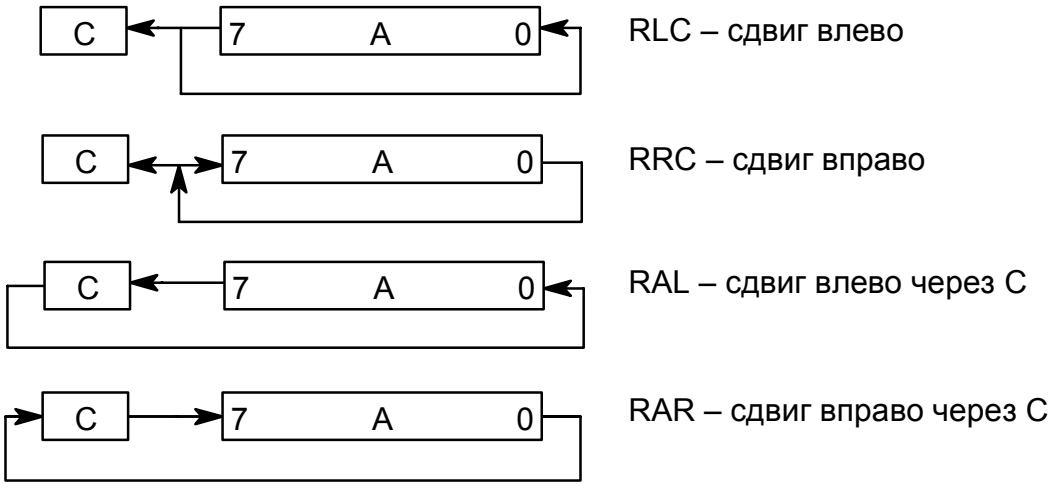


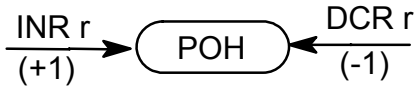
Рис. 7.16. Схема выполнения команд сдвига

7.8.4. ОПЕРАЦИИ В РОН И ПАМЯТИ

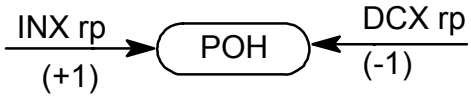
Ряд операций в МП комплекте КР580 могут быть выполнены помимо аккумулятора, непосредственно в РОН и ячейках памяти. К таким операциям относятся:

Инкремент/декремент

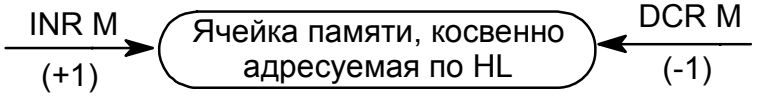
Арифметические одноопернадные операции увеличения на 1 (инкремент) и уменьшения на 1 (декремент) являются распространенными операциями для организации счетчиков при просмотре таблиц. На рис. 7.17 представлены примеры выполнения этих команд с РОН, регистровой парой и ячейкой памяти, косвенно адресуемой по HL.



r = A|B|. . .|L



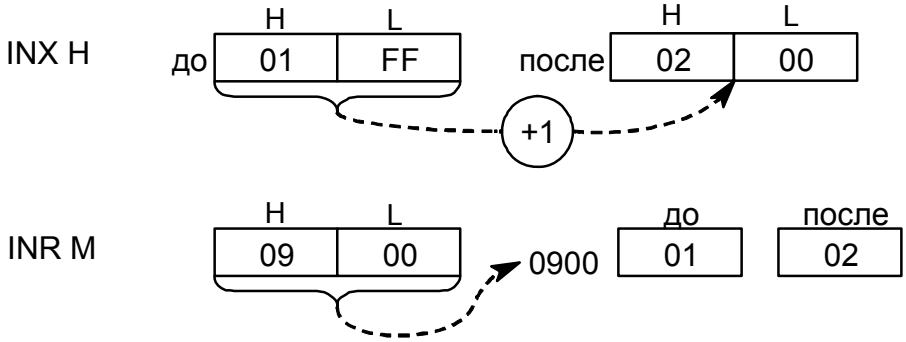
rp = B|D|H|L



РОН – регистр общего назначения

Рис. 7.17. Инкремент / декремент

Рассмотрим два примера:



Следует помнить, что при инкременте/декременте регистровой пары содержимое рассматривается как единое 16-разрядное число. При выполнении этой команды РгП не изменяется. При выполнении команды инкремента/декремента РОН и ячеек памяти, косвенно адресуемых по HL, РгП изменяется.

Сложение двухбайтовых операндов

DAD rp HL ← (HL) + (rp)

rp = B| D| H| SP

В РгП изменяется только признак С (CY).

7.8.5. КОМАНДЫ УПРАВЛЕНИЯ

Команды этой группы *не изменяют* содержимого РгП (F).

Команды безусловного перехода

По прямому адресу

JMP @, где @ – двухбайтовый адрес (@ → PC).

Например: JMP 0A01 ➤ адрес перехода

По косвенному адресу

PCHL – адрес перехода хранится в регистровой паре HL. При ее выполнении (HL) → PC.

Команды условного перехода

Jcon @, где con – мнемоника условия от английского слова condition.

Ранее отмечалось, что в качестве условия перехода используют состояние разрядов (флажков) РгП (F). Мнемоника, соответствующая этим состояниям, представлена на рис. 7.18.

con	NZ	Z	NC	C	P0	PE	P	M
усл.	Z=0	Z=1	C=0	C=1	P=0	P=1	S=0	S=1
	не нуль	нуль	нет переноса	перенос	нечет. кол-во 1	четн.	+	—

Рис. 7.18. Регистр признаков РгП(F)

Например: JC 8BFE – при C=1 переход по адресу 8BFE, при C=0 выполняется следующая по адресу команда.

Команды вызова ПП и возврата

Ранее отмечалось, что адрес возврата автоматически сохраняется в стеке, т.е. (PC) → стек.

Безусловные команды

CALL @ – вызов подпрограммы;

RET – возврат из подпрограммы.

Условные команды

Ccon @ – вызов подпрограммы;

Rcon – возврат из подпрограммы.

Действие команд аналогично действию команд условного перехода, т.е. если условие истинно, то вызов или возврат. Если нет, то выполняются следующие команды.

Прочие команды управления

RST n , где n = 0,1,...,7 – рестарт по вектору прерывания n.

При выполнении этой команды происходит передача управления подпрограмме, обслуживающей данное прерывание. В процессе выполнения команды RST содержимое счетчика команд PC запоминается в стеке, а в PC записывается адрес соответствующего вектора прерывания.

Этот адрес задается следующим образом. Команда RST имеет структуру 11NN N111, т.е. один байт. Трехразрядная комбинация NNN задается значением n

(n = 0...7). В счетчик команд PC заносится значение 0000 0000 00NN N000, которое служит адресом соответствующего вектора прерывания.

Таким образом, задавая определенное значение n, можно сформировать адрес одного из восьми векторов прерывания. Эти адреса располагаются в зоне от 0000H до 0038H адресного пространства и идут через 8 байт, т.е. под них зарезервированы первые 64 ячейки памяти (каждому из 8- векторов отведено по 8 байт). В этих зонах (по 8 байт) записывают только команды перехода к соответствующим подпрограммам (обработчикам), которые располагаются в других областях памяти.

Прерывающие подпрограммы (как и обычные подпрограммы) обязательно заканчиваются командой RET. В процессе выполнения этой команды адрес команды основной программы, перед которой произошло прерывание, выбирается из стека и передается в регистр адреса PA, а увеличенное на 1 значение заносится в счетчик команд.

EI – разрешение прерывания. Эта команда ставится в начале участка программы, на котором разрешено прерывание. По этой команде триггер разрешения прерывания в УУ МП устанавливается в состояние 1.

DI – запрещение прерывания. Эта команда ставится в конце участка программы, на котором разрешалось прерывание, и сбрасывает триггер в состояние 0.

NOP – "пустая" команда. Осуществляет пропуск четырёх тактов. Изменяется только PC.

HLT – останов. Вызывает прекращение выполнения программы и переход в состояние останова. МП отключается от внешних шин адреса и данных (т.е. их буферы переходят в состояние Z). На выходе WAIT (ожидание) устанавливается уровень 1. Это состояние может быть прервано сигналами запуска МП либо переводом его в состояние прерывания.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Какие устройства необходимы для создания простейшей микроЭВМ?
2. Перечислите 5 вариантов структур микроЭВМ.
3. Использование промежуточного интерфейса.
4. Что включает в себя понятие "контроллер ПУ"?
5. Перечислите характеристики процессора I8080.
6. Регистры данных. Их назначение.
7. Регистры признаков. Какие признаки хранятся в этих регистрах?
8. Опишите принцип двунаправленного обмена данными между внутренней и внешней ШД.
9. Какими регистрами программист может пользоваться?
10. Приведите структурную схему микроЭВМ на базе МП КР580ВМ80.
11. Из каких тактов состоит машинный цикл?
12. Перечислите форматы данных МП КР580ВМ80.
13. Перечислите форматы команд МП КР580ВМ80.
14. Какие способы адресации используются в МП КР580ВМ80?
15. На какие группы можно разделить команды МП КР580ВМ80?
16. Пересылки однобайтовые. Приведите примеры команд из этой группы.
17. Пересылки двухбайтовые. Приведите примеры команд из этой группы.
18. Какие операции в аккумуляторе вы знаете.
19. Операции в РОН и памяти. Какие операции к ним относятся?
20. Перечислите команды управления.

КОНТРОЛЬНЫЕ ЗАДАНИЯ

1. На листах ответа должны быть указаны номер группы, фамилия студента и номер его варианта.

2. Номера вопросов выбираются студентом в соответствии с двумя последними цифрами в его зачетной книжке. В табл.7.1 a_{n-1} – это предпоследняя цифра номера, a_n – последняя цифра. В клетках таблицы стоят номера вопросов, на которые необходимо дать письменный ответ.

Номера вопросов Таблица 7.1

a_{n-1} \ a_n	0	1	2	3	4	5	6	7	8	9
0	1,5,9, 13,16	2,6,10, 14,17	3,7,11, 15,18	4,8,12, 13,19	1,9,11, 14,20	2,10,12 ,15,18	3,5,8, 13,17	4,6,7, 14,19	1,8,10, 13,17	2,5,7, 14,18
1	3,6,8, 15,19	4,7,9, 13,16	1,5,12, 14,20	2,6,11, 15,16	3,7,10, 13,17	4,8,9, 14,18	1,7,12, 13,18	2,8,10, 14,19	3,5,11, 15,20	4,6,9, 13,17
2	2,7,9, 13,20	1,8,11, 15,19	4,5,10, 15,17	3,6,12, 14,16	1,5,9, 13,16	2,6,10, 14,17	3,7,11, 15,18	4,8,12, 13,19	1,9,11, 14,20	2,10,12 ,15,18
3	3,5,8, 13,17	4,6,7, 14,19	1,8,10, 13,17	2,5,7, 14,18	3,6,8, 15,19	4,7,9, 13,16	1,5,12, 14,20	2,6,11, 15,16	3,7,10, 13,17	4,8,9, 14,18
4	1,7,12, 13,18	2,8,10, 14,19	3,5,11, 15,20	4,6,9, 13,17	2,7,9, 13,20	1,8,11, 15,19	4,5,10, 15,17	3,6,12, 14,16	1,5,9, 13,16	2,6,10, 14,17
5	3,7,11, 15,18	4,8,12, 13,19	1,9,11, 14,20	2,10,12 ,15,18	3,5,8, 13,17	4,6,7, 14,19	1,8,10, 13,17	2,5,7, 14,18	3,6,8, 15,19	4,7,9, 13,16
6	1,5,12, 14,20	2,6,11, 15,16	3,7,10, 13,17	4,8,9, 14,18	1,7,12, 13,18	2,8,10, 14,19	3,5,11, 15,20	4,6,9, 13,17	2,7,9, 13,20	1,8,11, 15,19
7	4,5,10, 15,17	3,6,12, 14,16	1,5,9, 13,16	2,6,10, 14,17	3,7,11, 15,18	4,8,12, 13,19	1,9,11, 14,20	2,10,12 ,15,18	3,5,8, 13,17	4,6,7, 14,19
8	1,8,10, 13,17	2,5,7, 14,18	3,6,8, 15,19	4,7,9, 13,16	1,5,12, 14,20	2,6,11, 15,16	3,7,10, 13,17	4,8,9, 14,18	1,7,12, 13,18	2,8,10, 14,19
9	3,5,11, 15,20	4,6,9, 13,17	2,7,9, 13,20	1,8,11, 15,19	4,5,10, 15,17	3,6,12, 14,16	1,5,9, 13,16	2,6,10, 14,17	3,7,11, 15,18	4,8,12, 13,19

Учебное издание

Хмелевский Игорь Васильевич
Битюцкий Валерий Петрович

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

ОДНОПРОЦЕССОРНЫЕ ЭВМ

ЧАСТЬ 2

Редактор издательства *И.Г.Южакова*

Компьютерный набор авторский

Подписано в печать 20.06.2005		Формат 60 x 84 1/16
Бумага типографская	Офсетная печать	Усл. печ. л. 5,46
Уч.-изд. л. 7,8	Тираж 150 Заказ	Цена «С»

Редакционно-издательский отдел ГОУ ВПО УГТУ-УПИ
620002, Екатеринбург, ул. Мира, 19

Ризограф НИЧ ГОУ ВПО УГТУ-УПИ
620002, Екатеринбург, ул. Мира, 19