Микропроцессор К1801 ВМ1

Автор неизвестен. Текст обнаружен в дебрях интернета и приведен в приемлемый вид Сергеем Вакуленко. Исправлялись только синтаксис и пунктуация. Авторский ход мысли и стиль изложения тщательно сохранены.

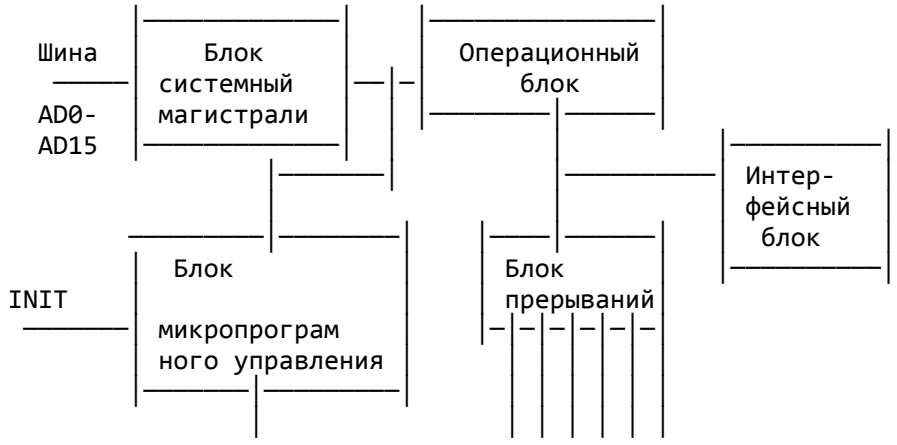
Микропроцессор К1801 ВМ1 представляет собой однокристальный 16-разрядный модуль. Этот конструктивно и функционально законченный модуль реализует систеу команд микроЭВМ “Электроника 60”, обспечивает управление системным интерфейсом, аналогичным по составу управляющих сигналов и алгоритмам обмена системному интерфесу микроЭВМ.

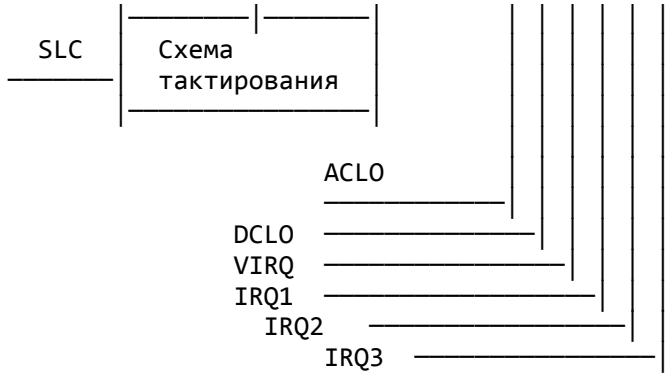
16-разрядный операционный блок включает в себя арифметическо-логическое устройство, секцию из 8 регистров общего назначения, регистр состояния процессора. Операционный блок осуществляет формирование адресов команд и операндов, прием данных с внутренней магистрали, выполнение арифметических и логических операций, хранение операндов и результата, выдачу в канал, формирование состояний процессора и адресов векторов прерывания.

Секция регистров общего назначения содержит восемь 16-разрядных регистров, которые могут выполнять функции регистров хранения данных и адресов, индексных регистров, регистров автодекрементной и автоинрементной адресации, указателей стека. Регистры R6 и R7 являются специализированными. Регистр R6 используется как указатель стека SP, а R7 — только как счетчик команд PC.

Кроме восьми программо-доступных регистров общего назначения, операционный блок включает в себя буферные регистры адресов данных, через которые осуществляется связь с внутренней магистралью процессора. Арифметическо-логическое устройство операционного блока формирует ряд признаков, записываемых в регистр состояния процессора. Признак I/O определяет приоритет выполняемой программы по прерыванию внешних устройств. Прерывание программы со стороны внешних устройств запрещено, если значение разряда 7 регистра состояния процессора равно 1. С целью упрощения отладки программы в любой ее точке командой записи слова состояния процес сора в T-разряд может быть записана 1. Это приведет к прерыванию программы с адресом вектора прерывания 14 (переход на программу связи с оператором). Остальные разряды регистра состояния процессора хранят признаки результата операции в арифметическо-логическом устройстве, используемые в командах ветвления программы. Установка признаков производится в следующих случаях: N=1, если результат отрицателен; Z=1, если результата равен 0; V=1, если при выполнении арифметической операции произошло переполнение; C=1, если произошел перенос из старшего разряда арифметическо-логического устройства, сдвиг единицы из старшего разряда влево или из младшего разряда вправо. Слово состояния процессора может быть считано и вновь записано в регист состояния процессора с помощью спциальных команд. При выполнении команд передачи управления содержимое регистра состояния сохраняется в стеке.

Структурная схема:





Обозначения на структурной схеме:

- AD0-AD15 — шина “Адрес/Данные”
- INIT — Инициализация вычислительного устройства, сброс тригеров (устройства с двумя устойчивыми состояниями) запросов прерываний.
- CLC — Сигналы тактовой частоты.
- ACLO — Авария сетевого питания
- DCLO — Авария источника питания
- IRQ1 — Сигнал перехода в пультовый режим
- IRQ2 — Сигнал прерывания с вектором (100) -восьмеричное.
- IRQ3 — Сигнал прерывания с вектором (270)-восьмеричное.
- VIRQ — Запрос прерывания от вычисоительного устройства.

В большой интегральной схеме К1801 ВМ1 можно выделить следующие фукциональные блоки, которые изображены на структурной схеме выше:

- 16-разрядный операционный блок обеспечивает формирование адресов команд и операндов, выполнение арифметических и логических операций, временное хранение операндов и результатов вычислений.
- Блок микропрограмого управления предназначен для выработки последовательности микрокоманд, соответствующий выполняемой команде.

Расмотрим устройство микропрограммного управления с использованием комутирующих схем в виде программых матриц. В данном случае последовательность выполнения этапов каждой команды задается при помощи программной матрицы. Кроме сигнала операции, матрица формирует импульсы, управляющие приемом, суммированием, сдвигом, выдачей и другими этапами, из которых складывается выполнение арифметических и логических операций. Таким образом, матрица совмещает функции комутатора операций и схем управления отдельными устройствами электроной вичислительной машины.

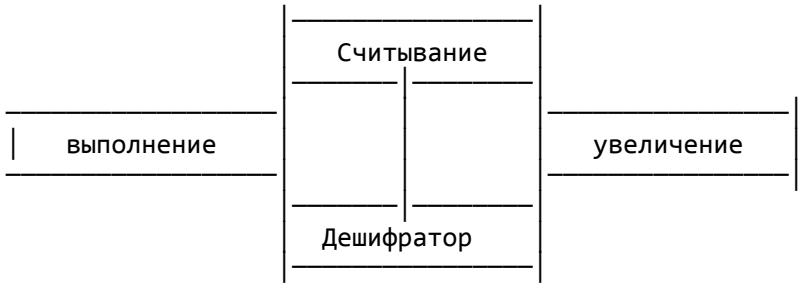
Устройство макропрограммного управления более универсально, чем микропрограммное, и позволяет вычислять адреса слов в процессе решения задачи и модифицировать структуру команды вычислительной машины. Генератор импульсов пульта управления задает темп работы устройств машины. Коммутатор сигналов программного управления, управляемый дешифратором операций и генератором импульсов, обеспечивает временную диаграмму работы машины. Регистр команд принимает, хранит и выдает код команды: признак, операция, адрес. Модификатор структуры команды управляет соответствующими командами и обеспечивает требуемое распределение команды между схемами устройства управления. Для паралельного выполнения команд используется несколько регистров команд. Регистр команд связан с магистралью слов, пультом управления, дешифратором операций, регистром индекса, счетчиком адресов команд и сумматором исполняемого адреса. Счетчик адресов команд увеличивает текущий адрес, выбираемый из памяти машины, на 1 или фиксирует новый исходный адрес при условных переходах, счечик связан по входу с комутатором переходов, который управляет дешифратором операций и регистром признаков переходов: знак, нуль, переполнение, абсолютная величина. По выходу счетчик связан с суматором исполняемого адреса, который осуществляет сложение исходного адреса с модифицирующим кодом, получаемым от регистра индекса. Последний связан по входу с суматором и магистралью слов; регистр управляется модификатором структуры и дешифратором операций.

Устройство макропрограмого управления работает по циклам, определяемым выполняемями командами. Цикл начинается подготовительными сигналами, приводящими схемы вычислительной машины в исходное состояние. После этого из памяти выбирается команда по адресу, сформированному счетчиком адресов команд и модифицированному сумматором исполняемого адреса. Команда, выбранная из памяти, фиксируется на регистре

команд. Дешифратор операций вырабатывает серию сигналов, обеспечивающих выполнение операции, которая задана кодом операции команды. Дешифратор адреса устройства памяти преобразовывает исполняемый адрес в сигналы, осуществляющие обращение к заданной ячейке памяти.

Блок прерываний обеспечивает работу приоритетной системы прерыванний, производя прием и предварительную обработку внутренних и внешних запроосов прерываний.

Цикл выполнения команды:



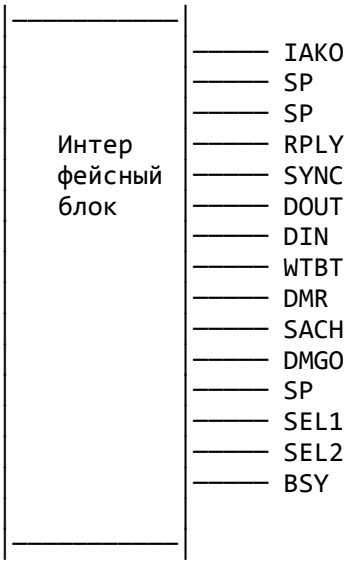
Система прерываний опрашивает состояние клавиатуры, экрана, дисковода. Выполняется приоритет выполнения операций.

Интерфейсный блок обеспечивает обмен информацией между микропроцессором и устройствами, подлюченными к системному интерфейсу.

Блок системной магистрали связывает внутреннюю магистраль микропроцессора с шиной “Адрес/Данные” системного интерфейса.

Схема тактирования обеспечивает сиснхронизацию всех функциональных блоков микропроцессора.

Интерфейсный блок организует обмен управляющей информацией между микропроцессором и устройствами, подключенными к системной магистрали, осуществляет арбитраж при операциях прямого доступа к памяти, формирует сигналы управления конвейером выборки кодов команд, адресов, данных. Связь между устройствами, подлюченными к системной магистрали, осуществляется по принципу активный-пасивный. В любой момент времени только одно устройство может быть активным. Оно осуществляет захват магистрали, укравляет циклами обращения к каналу, контролирует предоставление прямого доступа к памяти, разрешает прерывания от внешних устройств. Пассивное устройство формирует ответные сигналы, принимая и передавая информацию под управлением активного устройства.



Обозначения в схеме интерфейсного блока следующие:

- IAKO — Предоставление прерываний
- SP — Резерв

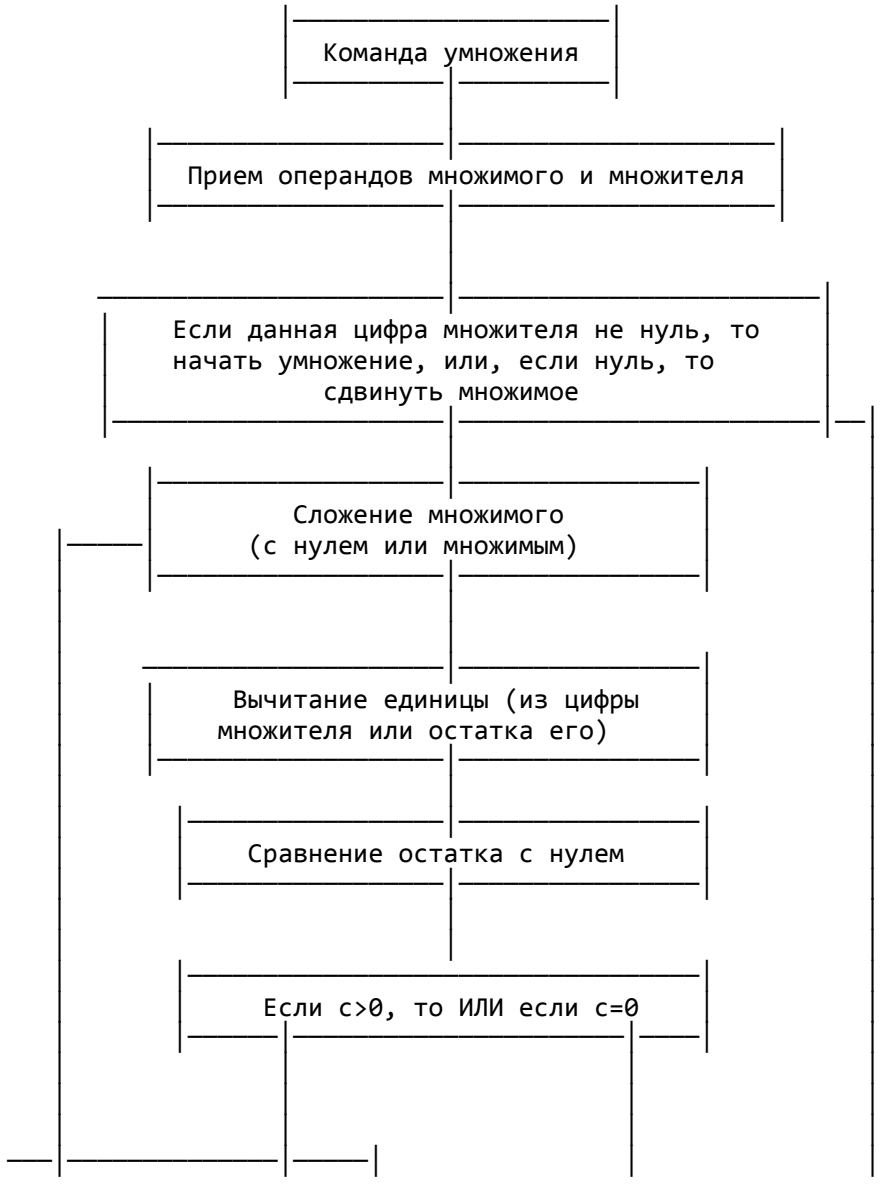
- RPLY — Ответ ведомого устройства
- SYNC — Синхронизация обмена (подтверждение выдачи адреса)
- DOUT — Вывод (запись)
- DIN — Ввод (чтение)
- DMR — Запрос прямого доступа к памяти
- SACK — Подтверждение предоставления прямого доступа к памяти
- DMGO — Предоставление прямого доступа к памяти
- SEL1, SEL2 — Выборка регистров ввода-вывода
- BSY — Сигнал занятости системного интерфейса

Работа процессора

Работу процессора можно уяснить, рассматривая автоматизацию решения математических задач. Общими случаями автоматического решения математических задач являются алгоритмы умножения и деления. Первый основан на сложении и сдвиге, а второй — на итерационном вычитании. Ограничимся подробным расмотрением процесса автоматического умножения. Схемный алгоритм, описывающий работу вычислительного устройства, дает исчерпывающий ответ на вопросы о синтезе электронных цифровых машин, так как точно отбражает процесс автоматизации вычислений.

Процесс вычисления распадается: при умножении — на циклы умножения на одну цифру множителя, а при делении — на циклы для определения одной цифры частного. Каждый цикл, в свою очередь, распадается на повторяющиеся этапы, состоящие из отдельных сложений (или вычитаний); при этом в случае деления число этапов заранее неизвестно.

Формальныйй характер вычислительного процесса показан на схеме автоматического умножения.



Таким образом, автоматическая вычислительная машина, реализующая алгоритм умножения, должна содержать: два приемных регистра, сумматор со сдвигом, регистр результата, счетчик и комутатор управления (схема сравнения, переключатель).

Для реализации алгоритма деления необходимы те же цифровые схемы, что и для умножения, а именно: два приемных регистра, сумматор со сдвигом, регистр результата, счетчики и схема сравнения (управления).

Из алгоритма видно, что процессор для выполнения умножения должен реагировать на цифры множителя в зависимости от величины остатка и переходить к тому или иному продолжению хода операции. Если остаток положительный, вычисление продолжается, если остаток отрицательный, то производится сложение и сдвиг, после чего осуществляется переход к следующему циклу вычитаний.

В практических задачах алгоритм обычно распадается на циклы повторяющихся серий элементарных операций, которые могут быть разбиты на более мелкие циклы.

Таким образом, программа автоматического решения задачи методом итерации должна предусматривать проведение последовательности вычисления этапов А или В, и в завимости от знака величины а (критерий точности) возможны два дальнейших хода процесса. В одном случае вычисления прекращаются, а в другом — производится переход к этапу С (замена исходных данных), после чего происходит возврат к повторению процесса с новыми исходными данными.

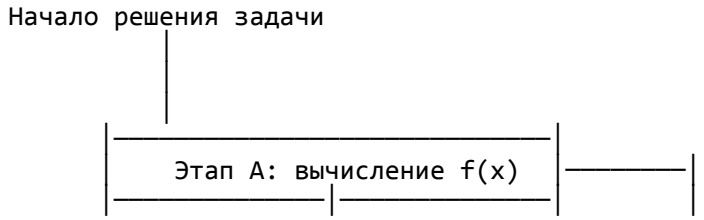
Для того чтобы иметь возможность осуществить такую программу вычислений, процессор должен реагировать на знак числа а, определяя то или иное продолжение прооцесса (появление того или иного знака должно вызывать ту или иную коммутацию). Далее должно быть предусмотрено возвращение от определенного этапа программы к более раннему.

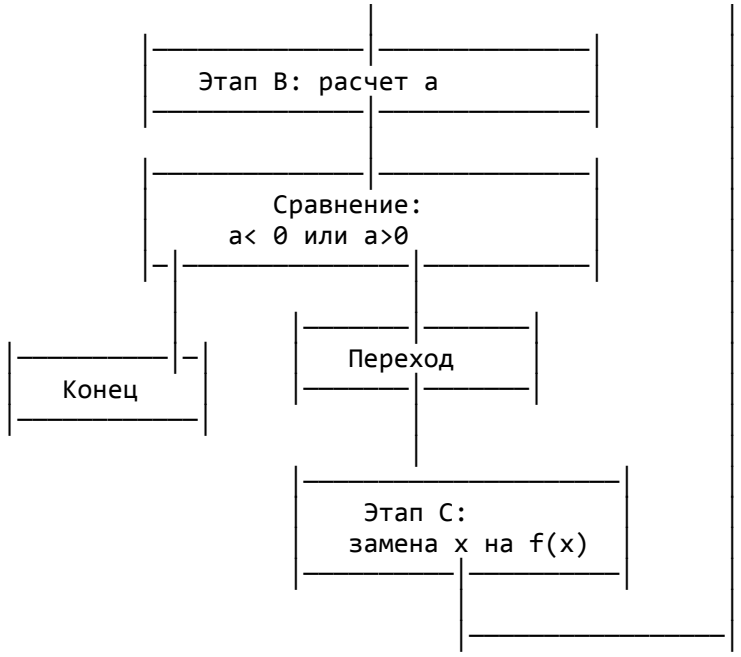
Это дает возможность реализовать решение задач, распадающихся на циклы.

Эфективность работы процессора зависит от ряда факторов, и на оценку ее влияют:

1. Технические параметры: быстодействие логических схем, время обращения к памяти, емкость памяти, совмещенность времни работы отдельных логических устройств во времени.
2. Тип обработки информции: последовательное выполнение программы.
3. Использование стандартных языков программирования.
4. Класс решаемых задач.
5. Участие оператора в работе процессора.

Схема проведения итерации:





Приближенную оценку производительности процессора можно получить по времени выполнения отдельных операций: сложение, сравнение и т. д.

Несколько лучшую оценку дает использование смешанного критерия, учитывающего относительное число операций различного типа для задая того или иного класса. Качество процессора при этом оценивается параметром:

```
P = n * t
```

где t - время выполнения команды типа данной машины; n - относительное число команд типа в программе данной задачи.

Время выполнения команды является характеристикой машины, а относительное число их — характеристикой класса задач.

Таким образом, параметр P характеризует время выполнения средней операции для данного типа задач. При этом емкость памяти, длина слова, структура команд, каналы ввода-вывода и внешние устройства не принимаются во внимание. По этой причине данная оценка характеризует только быстродействие вычислительного устройства с программным управлением, и отчасти — время обращения к памяти и систему команд. В этой связи можно сказать, что частота процессора БК модели 0011М составляет 6.3 Мгц. Получена на программе частотная характеристика радиоэлементов в 1993 г. Вообще говоря, частота процессора зависит от объема загруженой программы. С увеличением уменьшается частота достигая минимальной величины 4 Мгц, а при уменьшении увеличивается частота до 8 Мгц. (Смотри справочник по интегральным микросхемам).

Иногда оценка производительности вычислительной машины производится по времени решения некоторых типовых задач: формирование и сортировка массивов, обращение матриц, вычисление функций. Здесь трудности возникают из-за приближенного учета программых средсв.

Класификация и элементная база микропроцессоров

Микропроцессором называется программно-управляемое устройство для обработки информации и данных, реализованное в виде одной или нескольких больших интегральных схем. Программирование микропроцессора осуществляется подачей внешних электрических сигналов, комбинация которых образует определенную микрокоманду, обеспечивающую выполнение той или иной операции или микрооперации. Микропроцессоры делятся на несколько классов в зависимости от особенностей их структуры и значений основных параметров.

Способы программирования и организация управления.

В завимости от способа программирования различают микропроцессоры, выполняющие определенный набор команд и микропроцессоры, выполняющие набор микрокоманд.

Микропроцессоры первого типа выполняют набор из нескольких десятков (обычно 40-80) относительно простых команд. Для сравнения отметим, что современные мини-ЭВМ выполняют около 355 команд, а БК-11 только 64 команды. Реализация поступившей в микропрцессор команды обеспечивается с помощью блока управления, который вырабатывает необходимую последовательность управляющих сигналов-микроприказов, определяющих работу каждого функционального блока микропроцессора. В каждом такте машинного времени блок управления формирует определенную совокупность микроприказов — микрокоманду, в соответствии с которой различные узлы и блоки микропроцессора выполняют необходимые операции по поступающей информации. За несколько тактов выполняется ряд команд, необходимых для реализации поступившей команды. Отосительно простые команды реализуются за 2-5 тактов, более сложные требуют 10 и более тактов.

В микропроцессоре этого типа обычно используется так называемое аппаратное управление, при котором необходимая последовательность команд формируется с помощью специальных последовательных схем — управляемых генераторов чисел. При этом структура блока управления имеет вид, изображенный на схеме ниже.

Двоичный код команды, поступающий на входы микропроцессора, записывается в регистр команд и хранится там в течение времени ее выполнения. В соответствии с кодом дешифратором команд вырабатывается ряд управляющих сигналов. Часть сигналов дешифратора поступает в схему управления генератором чисел. Эта комбинационная схема задает последовательность чисел, вырабатываемых генератором, каждое из которых является кодом одной команды. В зависимости от кода поступившей команды схема управления определяет длину и состав последовательности чисел, формируемой генератором, т. е. число и виды команд, требуемых для выполнения данной команды. Вырабатываемые команды поступают на соответствуюшие узлы имкропроцессора, производящие необходимые операции над поступающими данными.

Сигнал, поступающий от дешифратора в счетчике тактов, определяет число тактов для реализации команды. После прохождения требуемого числп тактов счетчик вырабатывает сигнал окончания выполнения команды S, разрешающий переход к следующей команде программы, код которой хранится в одной из ячеек запоминающего устройства. Адрес этой ячейки определяется числом, содержащимсяя в счечике команд. При естественном ходе выполнения программы следующая команда содержится в ячейке памяти, адрес которой на единицу больше адреса хранения предыдущей команды. В этом случае адрес следующей команды образуют прибалением 1 к содержимому счетчика команд.

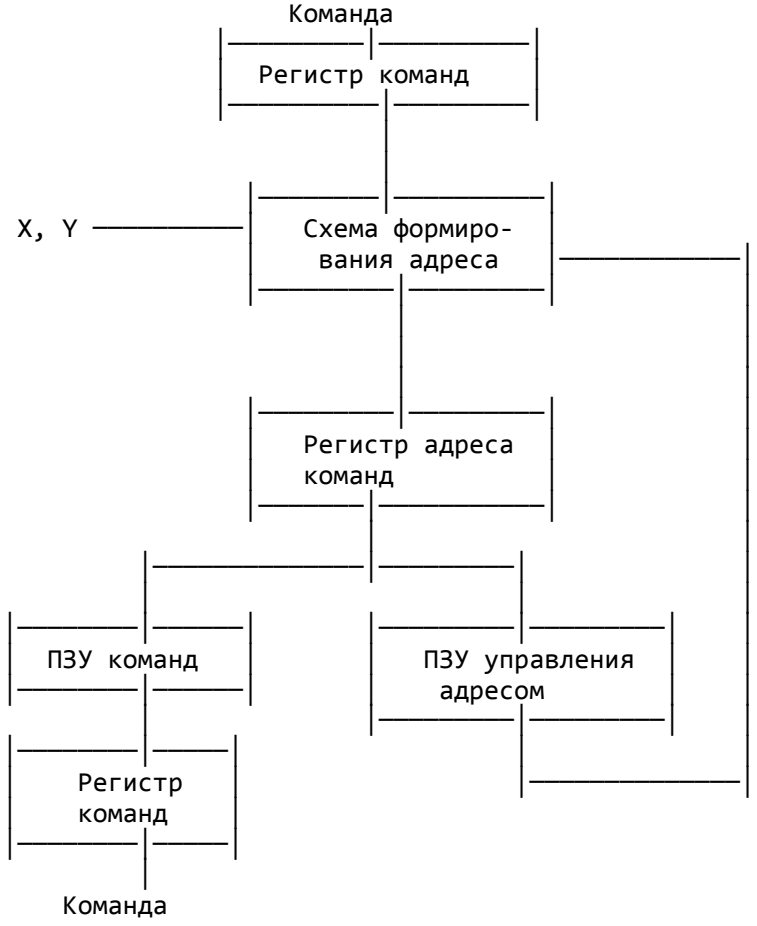
Схема аппаратного блока управления:

При выполнении команд условных и безусловных переходов естественный ход программы нарушается. В этих условиях схема управления счетчиком команд устанавливает его содержимое (адрес следующей команды) в зависимости от кода предыдущей команды (сигналов дешифратора) и результата его выполнения (сигналов от арифимитическо-логического устройства и других функциональных узлов). Сигналы, поступающие от микропроцессора во внешние устройства, могут потребовать прерывания выполнения одной программы и перехода к выполнению другой. Таким образом, сигналы также влияют на формирование команд и выбор следующей команды.

В микропроцессоре первого типа блок программного управления и операционные блоки, выполняющие операции над поступающими данными, содержатся на одном кристале. Такой микропроцессор представляет собой фукционально и конструктивно законченый блок обработки информации.

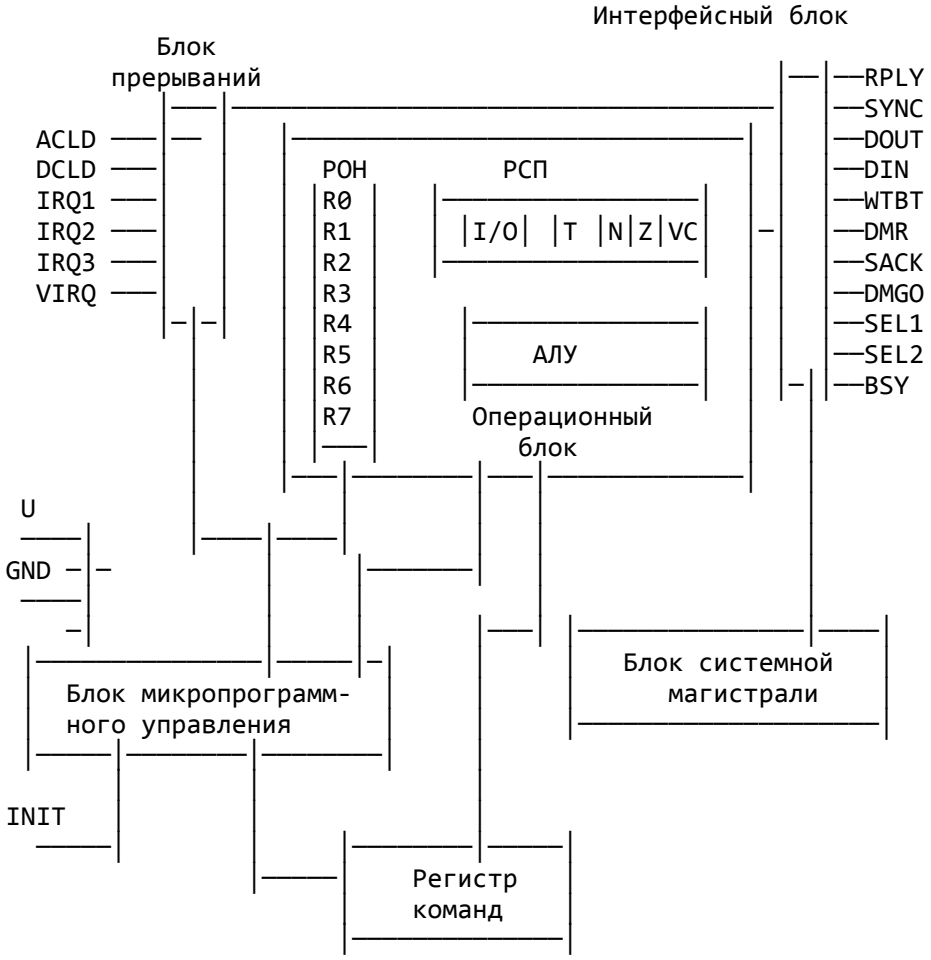
Микропроцессоры второго типа выполняют набор из нескольких десятков или сотен команд (обычно 64–256). Такие микропроцессоры можно использовать только в том случае, если программа работы задана в микропрограммном виде, т. е. в виде последовательности команд. Так как для реализации одной команды в среднем требуется выполнение 5–10 микрокоманд, то объем программ приблизительно на порядок превышает объем соответсвующих программ. Поэтому составление и отладка программ требует больших затрат времени. Обычно при использовании микропроцессора второго типа программа преобразуется к микропропрограммному виду автоматически с помощью специального микропрограммного управления, которое реализуется в виде отдельной специализированной большой интегральной схемы или строится из нескольких комбинационных и последовательных микросхем.

Структура микропрограммного блока управления:



Общая структура программных управляющих устройств показана на схеме выше. Коды всех команд хранятся в ПЗУ микрокоманд. Выбор требемой команды определяется ее адресом, который образуется с помощью схемы формирования адреса команд, функции которой в миропроцессоре обычно выполняются микросхемами. В соотвествии с кодом поступившей команды, хранящемся в регистре команд, схема формирует адрес первой из

последовательности команд, обеспечивающий реализацию поступившей команды. Выбранная их ПЗУ команда заносится в регистр команд и затем поступает в соотвествующие операционные блоки. Одновременно из ПЗУ управления адресом выбирается комбинация двоичных управляющих сигналов (код управления), который поступает в схему формирования адреса, вызывает обращение на ее выходе адреса следующей команды. Этот адрес и соответствует последовательности команд могут изменятся под воздействием внешних сигналов устройств и результатов предыдущей команды. Для выбора следующей команды также формируются необходимые команды, в соотвествии с которыим операционные блоки образуют новый адрес.



Выполнение любой команды микропроцессора связано с одним или несколькими обращениями к магистрали, выполняемыми по одному из трех циклов: ввод (чтение), вывод (запись), ввод-пауза-вывод (чтение-модификация-запись). Любой цикл начинается выставлением сигнала BSY — занятия на системной магистрали. Этот сигнал используется также для управленим схемами магистрали. Одновременно с сигналами WSY на магистрали выставляется адрес, а при обработке цикла “Запись” — также и сигнал WTBT. С задержной на такт выставляется сигнал SYNC (синхронизация обмена), свидетельствующий о том, что на выводах ADO-AD15 установлен код адреса. Для любой магистрали первое обращение к магистрали связано с выставлением на магистрали адреса из счтчика команд PC.

Во время действия сигнала SYNC микропроцессор вырабатывет сигнал DIN, свидельствующий о готовности микропроцессора принять данные от пассивных устройств. Сигнал DIN вырабатывается также совмемтно с сигналом IAKO, сопровождая ввод адреса вектором прерывания. Если микропроцессор выводит данные, то вместо DIN вырабатывается сигнал DOUT, свидетельствующий о том, что выводимые данные установлены на выводах блока системной магистрали. При выводах слова, сигнал WIBT снимается вместе с адресом, а при выдаче байта сохраняется до окончания сигнала BSY.

Сигнал RPLY вырабатывается внешним устройством в ответ на сигнал DIN или DOUT и обозначает в первом случае, что данные установлены на магистрали, а втором случае — что данные приняты с магистрали.

Если сигнал RPLY от внешего устройства не поступил в течение 64 тактов, то микропроцессор переходит к обслуживанию внутреннего прерывания по ошибке

обращения к системной магистрали. После поступления сиганла RPLY от внешего устройства микропроцессор считывает данные с магистрали и сбрасывает сигнал DIN. Если выполняется цикл “Запись”, то сигнал DOUT снимается через 300 нс и через 100 нс считываются данные с магистрали.

Внешнее устройство после сброса сигнала DIN или DOUT снимает сиганал RPLY, после чего микропроцессор снимает сигналы BSY и SYNC и может выполнять следующий цикл обращения к каналу.

Для увеличения общей производительности микропроцессора может отрабатываться совмещенный цикл “Чтение-модификация-запись”, в котором после снятия внешним устройством сигнала RPLY сигналы SYNC и BSY не снимаются, а отрабатывется временная диаграмма, соотвествующая циклу “Запись”.

В микропроцессоре имется спецефический режим обращения с внешними устройствами расширения ввода-вывода, задаваемый сигналами SEL1, SEL2. Сигналу SEL1 соответсвует адрес 177716, а сигналу SEL2 — адрес 177714. Обмен с этими регистрами осуществляется обычным образом по сигналам DIN и DOUT, однако выдачи от регистров ответного сигнала RPLY не требуется. По длительности сигналы SEL1 и SEL2 совпадают с сигналами BSY.

Для организации многопроцессорных систем и прямого доступа к памяти (ПДП) в интерфейстном блоке вырабатываются специальные сигналы DMR, DMGO, SACK. В основе этих функий лежит алгоритм захвата шины адрес/данные главным и подчиненным устройством. Неглавное устройство формирует сигнал DMR (”Запрос прямого доступа”), на который главное устройство вырабатывает сиганал разрешения DMGO. Все подчиненные усиройства в многопроцессорой системе соединены последовательно по шине захвата, так что выход DMGO предыдушего, соединяется со входом DMG1 последуещего. При этом сигнал DMGO транслируется со входа DMG1 на выход DMGO0 на всех микросхемах, кроме выставившей запрос DMK, которая блокирует дальнейшее прохождение сигнала DMGO и выставляет сигнал SACK (”Подтверждение запроса магистрали”).

После этого микропроцессор, захвативший магистраль, снимает сигнал DMR, выставляет адрес и сигнал BSY, после чего отрабатывает один из сигналов обмена. Сигнал SACK снимает одновременно с сигналом DIN или DOUT, если в данном микропроцессоре снова не выставлен запрос DMR. В этом случае запрос обращения к магистрали может быть повторен.

Главный микропроцессор может захватить магисталь только в том случае, если отсутствуют запросы от других устройств и не идет цикл обмена по магистрали. Цикл обмена начатый главным микропроцессором, при появлении других запросов будет завершен, но переход к новому циклу приостанавливается, а магистраль предоставляется подчиненому микропроцессору, выствившему запрос. Таким наивысшим приоритетом в системе будет обладать устройство, соединенное своим входом DMG1 с выходом DMG0 главного микропроцессора. Назначение микропроцессора в системе главным или подчиненым осуществляется заданием кода на входе PA1, PA0. Код 11 соотвествует главному микропроцессору. другие коды — подчиненному.

Важной функцей при организации управления в реальном режиме времени и построении микропроцессорных систем является прерывание. В качестве источников прерывания могут выступать следующие процессы и сигналы:

- ошибка обращения к магистрали;
- незадействованный код в регистре команд;
- T-бит в реггистре слова состояния процессора;
- сигнал аварии сетевого питания ACLO;
- сигналы радиальных прерываний IRQ1, IRQ2, IRQ3;
- сигнал векторного прерывания VIRQ.

Как указывалось выше, прерывания от внешнего устройства IQR1, IRQ2, IRQ3, VIRQ могут быть заблокированы установкой в 1 седьмого разряда слова состояния процессора. Как правило, прерывание текущего процесса производится после выполнения очередной команды. Исключение сотавляет ошибка обращения к магистрали, вызывающая прерывание програмы на любой фазе выполнения команды.

Процедура внешего прерывания начинается с подачи внешним устройством запроса VIRQ на соответствующий вход микропроцессора. Если прерывание не замаскировано (разряд 7

РСП равен 0), то микропроцессор по указателю стека R6, с предварительным декриментом адреса, записывает содержимое РСП, а затем счетчика команд РС. Кроме того, микропроцессор последовательно устанавливает сигналы DIN (управление вводом данных) и IAKO (разрешение прерывания). Сочетание этих сигналов при пассивном уровне сигнала SYNC обеспечивает отработку цикла ввода адреса вектора прерывания. Устройство, выставившее запрос, принимает сигнал IAKO и запрещает его распространение к другим устройствам, выставляет адрес вектора прерывания на системную магисталь, вырабатывает сигнал RRPLY и снимает сигнал VIRQ. Микропроцессор, приняв адрес вектора прерывания, снимает сигналы DIN и IAKO, и устройство заканчивает цикл снятием сиганала RPLY.

Далее микропроцессор считывает новое содержание РС и РСП из двух ячее памяти (адпес первой есть адрес вектора прерывания) и переходит к выполнению команды с адресом из РС, входящим в программу обработки прерывания. При выполнении команды также может быть реализовано прерывание, и снова состояние выполняемой программы будет зафиксировано в очередной ячейке стека.

Всякая программа обработки прерывания должна заканчиваться командой возврата из прерывания RTI, по которой из стека выбирался вектор прерванного процесса, записываемый в РС и РСП. Сигнал IRQ1 блока прерывания переводит микропроцессор в состояние, аналогичное состоянию команды HALT, и используется для контроля за положением переключателя “Программа/пульт”. Сигналы IRQ1, IRQ2, IRQ3 используются также для организации систем радиальных прерываний, имеют прериотет выше, чем VIRQ, и вызывают прерывание по фиксированным адресам 000100 и 000270 соотвественно.

В блоке прерывания имеются два входа: ACLO — сигнал аварии источника питания переменного напряжения, DCLO — сигнал аварии источника питания постоянного напряжения. Этот сигнал вместе с сигналом INIT блока микропрограммого управления используется для осуществления процедуры инициализации микропроцессора. В первый момент после включения питания микропроцессора поступают сигналы DCLO и ALSO. Сигнал DCLO должен выдерживаться не менее 5 мс, в течении которых микропроцессор вырабатывает сигнал INIT, используемый для установки в исходное состояние внешних устройств (сигнал INIT может также поступать в микропроцессор извне, осуществляя сброс тригеров запроса радиальных прерываний и блокировку сигналв DMR). Через 5 мс после подачи питания на микропроцессор должен быть снят сигнал DCLO (микропроцессор при этом снимает сигнал INIT); через 70 мс должен быть снят сигнал ACLO. С этого момента начинается собственно инициализация (обнуление) работы микропроцессора: по адресу 1777716 (для главного микропроцессора) считывается стартовый адрес SEL1 и заносится в старший байт счечика команд РС; в РСП загружается константа 340; микропроцессор переходит к выполнению команды по адресу из счетчика команд.

Одной из функций описанных выше сигналов интерфейсного блока является управление блоком системной магистрали, связывающей внутренную магистраль микропроцессора с внешней. В этом блоке осуществляется управление усилителями приема и выдачи информации по совмещенным выводам адресов и данных AD0-AD15.

Используемые в БК типы данных

Бейсик позволяет работать с данными арифметичекого и текстового (строкового) типов, причем арифметические данные могут быть целого и веществеого типов (одинарной и двойной точности). Конечно, процессор БК не имеет таких возможностей. Например, обработка вещественных чисел может быть организована только программно, причем различные системы программирования могут использовать даже различные способы представления этих чисел.

Основными данными, с которыми может работать процессор БК, является байт и слово. Соответственно, в слове может быть размещено целое число, а в байте — один символ текста.

В одном слове 16 бит — двоичных разрядов, поэтому из них можно составить 2^16 = 65536 различных комбинаций. Если учесть, что машинные слова команды работы с целыми числами считают старший разряд (15-й бит) слова знаковым (0-число положительное, 1-

число отрицательное), то диапазон чисел в БК будет 32767. Если в результате вычислений в Бейсике получится больший результат, он выдаст ошибку 6. Если то же самое произойдет в вашей программе в кодах, никто этого не заметит, если не принять специальные меры. Тогда, прибавив единицу к числу 32767, вы получите -32768.

Для того, чтобы представить символы в памяти машины, их кодируют. В одном байте 8 бит, что позволяет закодировать 2^8 = 256 различных символов. Этого вполне достаточно, чтобы разместить строчные и заглавные буквы двух алфавитов, цифры и другие спецзнаки. Именно для удобства кодировки символов и был выбран такой размер памяти машины.

Следует отметить, что для кодирования символов в БК используется советский стандарт KOI-8 (8 бит), базирующийся на международном стандарте ASCII (American Standard Code for Information Interchange).

Процессор — это сложная электронная схема. Однако знать все его особенности при программировании в машинных кодах незачем. То, что нужно знать программисту о каком-либо устройстве машины, называется его программной моделью. Сюда относятся программно-доступные ячейки памяти устройства (их называют регистрами), а также алгоритмы функционирования устройства.

Регистры общего назначения предназначены для промежуточных результатов вычислений. Операции пересылки данных в ОЗУ. Оптимальное импользование регистров общего назначения в чисто выполняемых чиклах программы позволяет иногда ускорить ее на порядок. Регистр R5 используется для связи с подпрограммами, а регистр R6 обозначается SP и используется как при вызове подрпрограм, так и при обработке прерываний. Регистр R7 обозначается PC — счетчик команд, всегда содержит адрес следующей команды, которую должен выполнить процессор. Расмотрим подробнее алгоритм работы процессора, то есть опишем как он выполняет программу.

Программа должна начинать работу с определенного адреса памяти (с опредeнной команды). Для этого двоичный код адреса, с которого начинается программа, должен быть предварительно записан в регистр PC процессором (счетчик команд), так как регистр PC всегда хранит адрес очередной команды, подлежащей выполнению.

Допустим, в регистре PC записали восъмеричное число 7000 (все коды и адреса будем писать в восьмеричном виде). Это значит, что следующей командой, исполняемой процессором, будет команда, хранящаяся по адресу 7000. Произойдет это так: из памяти с адресом 7000 в процессор перепишется слово — двоичный код команды для выполнения. Например, пусть по адресу 7000 была записана команда 600001.

По этой команде процессор должен сложить содержимое R0 с содержимым R1. Причем результат сложения (сумма) запишется в R1. После выполнения процессором этой команды содержимое регистра PC автоматически увеличивается на 2 (станет равным 7002), теперь уже в процессор будет передана команда из ОЗУ с адреса 7002, и так далее.

Действие команд перехода заключается в записи в регистр PC нового значения, и тогда процессор продолжит выполнять программу с указанного адреса.

А что будет, если процессор в результате ошибки программиста прочитает непонятную ему команду? Ничего страшного не произойдет, просто в этом случае выполнение программы будет прервано и процессор начнет выполнять программу из системного ПЗУ (как говорят, программа вылетит в монитор), либо управление возмет на себя используемый отладчик.

Аналогичная ситуация возникает, если в PC записать нечетное число (вы помните, что длина команды кратна машинному слову, т. е. двум байтам, и ее адрес должен быть четным), либо такого адреса в машине не существует.

Остается только добавить, что после выполнения каждой команды программы меняется содержимое регистра состояния процессора. Этот регистр предназначен для хранения PSW — слова состояния процессора.

В слове состояния процессора имеют значения следующие биты (разряды):

- бит 0 (C) устанавливается в 1, если при выполнении команды произошел перенос единицы из старшего разряда результата;

- бит 1 (V) устанавливаетсч в 1, если при выполнении арифметической команды (например сложения) произошло арифметическое переполнение;
- бит 2 (Z) устанавливается в 1, если результат равен нулю;
- бит 3 (N) устанвливается в 1, если результат отрицателен.

Эти биты слова состояния процессора используют команды условного перехода. Остальные биты слова состояния процессора устанвливаются программистом для задания режимов работы процессора:

- бит 4 (T) при установке в 1 вызывает после выполнения очередной команды прерывание по вектору 14. Это используется программами-отладчиками для трассировки программы;
- бит 7 (P) при установке в 1 запрещает прерывание от внешних устройств.

Таким образом, командой установки слова состояния процессора MTPS #200 можно запретить прерывание от клавиатуры до тех пор, пока не выполнится команда MTPS#0.

Управление памятью

Обычная интерпретация машинного слова как адреса позволяет нам адресовать к байтам в диапазоне от 0 до 17777 т. е. всего к 2^16 = 65536 байтам. При описании различных объемов памяти принято пользоваться обозначением К, которое соответствует числу 2^10=1024 (= O2000). Поэтому можно сказать, что одно слово машины позволяет вам адресовать к полю памяти 64 Кбайт или 32К слов.

В стандартном оборудовании каждого процессора предусмотрен определенный объем памяти, который в случае необходимости можно расширить с помощью имеющихся в апаратуре возможностей. Было естетвенно считать, что максимальный объем памяти для машины равен 32 К слов, просто потому, что для большего не хватает адресного пространства. Так как 4 К слов общей шины зарезервированы под регистры ввода-вывода и различные управляющие символы, то, казало сь бы, система предоставляет в распоряжение не более чем 28 К слов оперативной памяти. Для самых малых машин данного семейства это именно так.

Однако в боее крупных системах внутренние регистры процесора имеют 8 разрядов, так же, как и адреса общей шины. Поэтому процессор может адресовать, а общая шина обепечивать доступ к пространству в 2^18 = 256 К байт или 128 К слов. Учитывая 4 К слов, отводимых под регистры ввода-вывода и аналогичные срества, потенциально получаем 124 К слов оперативной памяти. Казалось бы, это огромный объем но некоторые пользователи (с непомерными запросами), работающие в системах с разделенным временем, умудряются его полностью исчерпать.

Указанная память, однако, составлена из уже имеющихся у нас шестнадцатиразрядных слов машины, и, казалось бы, проблема остается неразрешимой, поскольку одно слово из шестнадцати битов не может вместить восемнадцатиразрядный адрес. Если пользователи не принимают мер, то так оно и будет: только 32 К адресов общей шины будут доступны его программе. Из них 28 К — ячейки опративной памяти. Так как 56 К =O160000 (в байтах), то обычным способом можно программно послать на ячейки от 0 до 157776. Ссылка интерпретируется процессором обчным образом: как обозначение ячейки общей шины с указанием номера. К примеру, команда CLR @#100000 очищает ячейку общей шины с номером 100000. Эта ячейка является одним из слов оперативной памяти. В данном случае мы будем использовать абсолютную адресацию, поскольку значение адреса представляется при таком способе в явном виде.

Регистры устройства ввода-вывода, однако, всегда находятся в верхних адресах общей шины и занимают 4 К слов, т. е. в случае восемнадцатиразрядных адреслв это ячейки с 760000 по 777776. Например, буфер печатаюшего устройства занимает ячейку с адресом 777566. Такое число не помещается в одно слово машины, изза чего команду типа MOV @#102, 777566 закодировать без изменения знака нельзя. Тем не менее мы уже видели, что запись MOVB #102, @177566 позволяет адресоваться к требуемому буферу. Происходит так потому, что процессор автоматически смещает значения всех адресов в диапазон от 160000 до 177776, интерпретируя их как шестнадцать младших битов истинного адреса, в котором 17-й и 18-й биты равны нулю.

Очень важно не путать подобное смещение адресов с тем, которое производит компоновщик. Задача последнего состоит в том, чтобы вне зависимости от места размещения программы в памяти исполнительный адрес, вычисляемый процессором, соответствовал требуемой ячейке памяти. К примеру, если мы пишем MOVB #102, 177566, то от компоновщика требуется определенная работа, в результате которой он убедится, что здесь имеется ссылка на ту ячейку памяти, что и при записи MOVB #102, @#177566. В обоих вариантах ссылаются на ячейку 177566. В функции смещения, о которой идет речь в данном разделе, входит привязка адреса (виртуального адреса) к соответсвующей ячейке общей шине (физическому адресу). Обратите внимание также на то, чио преобразование виртуального адреса в физичесский выполняется аппаратно.

Ясно, что в диапазоне от 0 до 157776 никакого смещения не требуется: виртуальный адрес совпадает с физическим. Однако для виртуальных адресов с 160000 по 177 177776 физический адрес получается расширением виртуального адреса двумя единичными битами: разрядами 17 и 18.

Машины с памятью более чем 28 К слов имеют специальный блок проверки величины смещения. При соотвествующем подборе виртуального адреса и значения смещения любой физический адрес становится доступным. Апаратура, при помощи которой осуществляется контроль смещения, называется блоком управления памятью.

Страничная организация памяти

На машине пространство виртуальных адресов разбито на восемь страниц, каждая из которых рассматривается блоком управления памятью как единое целое. Имеется 32 К слов виртуальных или программных адресов, а длина каждой страницы равна 4 К слов. Привязка виртуальных адресов к страницам определена заранее и не может быть изменена. Конкретно она выглядит так:

Номер страницы	**Диапазон виртуальных адресов**
0	000000–017776
1	020000–037776
2	040000–057776
3	060000–077776
4	100000–117776
5	120000–137776
6	140000–157776
7	160000–177776

С каждой страницей связаны две специальные ячейки общей шины: регистр адреса страницы и регистр описания страницы.

Включение управления памятью

При введении в действие процессора или при перзапуске его после команды HALT блок управления памятью не работает. Он активизируется установкой бита включения управления памятью, а именно нулевого бита регистра состояния управления памятью номер 0; заметьте, что пока управление памятью выключено, автоматическое подлючение седьмой страницы позволяет нам обращаться к регистру R0, как к виртуальному адресу 177572. Поэтому включение блока управления памятью достигается так:

```
        R0=177572
        . . . .
        BIS #1, @#RO
```

Рассмотрим теперь такую последовательность команд:

```
        R0=177572
        R7=172356
        . . . . .
        CLS  R7
        BIS #1, @#R0
        BIC #1, @#RO
```

Если машина только что включена, то нет необходимости сбрасывать регистр R7, так как он заведомо содержит 0. Но если процессор остановлен после фукционирования операционной системы, то, несмотря на отключение блока управления памятью, регистры продолжают сохранять установленные в них системой значения. При нормальной работе блок управления памятью запомнит содержимое регистра R7, так что седьмая страница будет, как обычно, отведена под регистры устройств ввода-вывода. Для этого требуется специальным образом настроить упомянутый регистр, содержимое которого было вытеснено нашей командой.

Нужно помнить однако, что сбрасывание битов регистра производилось при отключенном блоке управления памятью. Поэтому ссылка в следующей проверке на регистр R0, как на виртуальный адрес 177572 аппаратно приводит к физической ячейке с адресом 77572. Следовательно, эфект от выполнения команды BIS сохраняется.

Следуещей командой BIC мы попытались отключить блок управления. Именно к такому результату привело бы сбрасывание первого бита ячейки 77572 общей шины. Но в нашей команде мы адресуем к 177572. Предыдущая команда привела к отключению автоматичсекой привязки вирутуального адреса 177572 к физической ячейке 77572. Поэтому аппаратура, заметив, что 177572 есть виртуальный адрес в седьмой странице, обратится к регистру R7, чтобы определить, как его надо преобразовать. Мы намеренно воздержались от записи в регистр R7 необходимого смещения. В итоге команда BIC обратится не к регистру состояния, а к некоторой другой физической ячейке памяти и не выполнит возложенную на нее фукцию.

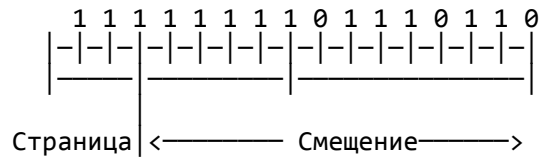
Не существует иного способа программно исправить сложившуюся ситуацию. Поскольку невозможно обратиться к регистру R7, нельзя задать и правильное смещение. По аналогичной причине все регистры устройства ввода-вывода оказываются программно не достижимы. Единственый способ отключить блок управления памятью состоит в останове процессора.

Формирование физического адреса

При включенном блоке управления памятью каждая ячейка, к которой адресуются из программы, расматривается аппаратурой как состоящая из двух полей. Биты с 13-го по 15-й определяют номер страницы виртуального адреса. Это так называемое поле текущей страницы. Пусть виртуальный адрес равен 177566. Биты с 13-го по 15-й в нем равны 1, что соотвествует двоичной записи числа 7, т. е. данный адрес задает седьмую страницу. Поэтому аппаратура обращается к регистру R7 за базовым адресом страницы 7. Он кодируется битами с 0-го по 11-й регистра адреса страницы (остальные биты не используются). Для получения действительного значения смещения это двенадцатиразрядное поле сдвигается на 6 битов влево.

Оставшиеся 13 битов виртуального адреса (с 0-го по 12-й) представляют собой поле смещения внутри данной страницы. Значение этого поля складывается с базовым адресом и в результате формируется физический адрес, соотвествующий данному виртуальному.

Обратите внимание на то, что все действия выполняются аппаратно. Программисту только необходимо только настроить регистры. Обычно операционная система заносит в регистр R7 число 7600. Пусть опять витуальный (программный) адрес равен 177566. Он распадается на два поля:



которые определяют седьмую страницу и смещение 17566. Отметим, что смещение всегда задается (в байтах) между 0 и 17777, так как длина страницы составляет 8 К (=О2000) байтов или 4 К **слов**. Смещая содержимое регистра R7 (т. е. 7600) на 6 битов влево, получим базовый адрес данной страницы, равный 76000. Следовательно, физическим адресом, соотвествующим данному виртуальному, будет:

```
    +  Базовый адрес        76000
       Смещение              1756
_____________________________________
       Физический адрес     87756
```

Страница виртуальной памяти всегда содержит 200 квантов. Порция в 100 байтов есть наименьшая величина смещения, и поэтому в блоке управления памятью она служит квантом. Блок управления памятью распределит первый квант страницы, вычислив базовый адрес по описанной выше схеме. Но он не будет выполнять ту же операцию с остальными квантами до особой команды. В результате адресация к ячейкам 000100 по 017776 не достигнет цели, поэтому что им не будут поставлены в соответствие реальные ячейки памяти. Ссылки же на адреса с 0200000 по 020076 будут реализованы.

Чтобы задать количество квантов данной страницы виртуальных адресов, которое должно быть приведено к физическим ячейкам памяти, необходимо обратиться к регистру описания страницы. В его старший байт программист заносит требуемое число квантов (помимо первого). По команде:

```
        BIS #77400, @#R7
```

блок управления памятью узнает о том, что все 200 квантов седьмой страницы должны преобразовываться в физический адрес. Команда задает размер страницы в 200 квантов.

Первый и второй байты регистра данных каждой страницы должны усианавливаться программистом, чтобы определить права доступа процессора к ней. Если они оба равны 1, то разрешаются любые обращения из прогарммы к соответствующей виртуальной памяти; это резидетная, открытая на чтение-запись страница. Если в первом бите 1, а во втором 0, то ссылки разрешены только в командах, которые не изменяют содержимаое ячеек памяти; это резидентная, открытая только на чтение страница. Если наконей оба бита нулевые, програ мная адресация к данной странице запрещена; это нерезидентная страница.

Таким образом, последовательность команд:

```
        MOV #7600, @#R7
        BIC #6, R7
        BIS "1, @R0
```

позволяет эффективно прервать связь процессора с устройствами ввода-вывода (если только к ним уже не адресовались из другой страницы). Действительно, хотя в регистре R7 установлено правильное смещение для седьмой страницы, сбрасывание битов командой BIC делает страницу нерезидентной. Поскольку все регистры управления памятью расположены в одной и той же области общей шины, теперь невозможно выделить под нее другую страницу. Снова единственный выход — остановить и перезапустить процессор.

Заметьте, что замена #6 на #77400 в команде BIC дает тот же эфект. Только один квант седьмой страницы будет привязан к физической памяти, а все адреса выше 76076 окажутся недоступными.

Понятно, что под регистры ввода-вывода можно приспособитть произвольно выбранную страницу. Для этого достаточно настроить все регистры. Так как они находятся в одном и том же месте памяти, то предварительно придется либо отключить управление памятью, либо получить к ним доступ через другую страницу.

Напишем программу вывода на терминал буквы В, на этот раз адресуясь к буферу печати, как ячейке памяти с виртуальным адресом 34566. Виртуальный адрес рападается на поле с номером 1 и на поле смещения 1456. Следлвательно, чтобы виртуальный адрес 34566 соответствовал физическому адресу 77566, необходимо в R1 задать базовый адрес, который определяется так:

```
        Физический адрес          77566
      - Смещение                - 1456
  ------------------------------------
        Базовый адрес             76110
```

И, следовательно, в R7 нужно занести значение 76110. Вся программа, на момент входа в которую предполагается, что блок управления памятью отключен, такова:

```
          R1=17232
          R0=17572
  START:  MOV #76110, @#R1
          MOV #1456, @#R1
          BIS #1, @#R0
          MOVB #102, @#34566
          HALT
          .END
```

Защита памяти

Ограничения на размеры страницы и права доступа к ней, которые можно установить в регистре описания, используются в системах с разделением времени для того, чтобы предохранить выделенные различным пользователям пространва памяти от пересечения и чтобы защитить операционную систему от всех непривилегированных программистов.

Если в команде нарушается какое-либо их этих ограничений, она выполняться не будет: блок управления памятью игнорирует ее. Он вызовет программное прерывание (невзирая на приоритет процессора) с вектором прерывания в ячейке 250. В связи с этим надо учесть, что для всех векторов внешних и программных прерываний используются виртуальные адреса. Поэтому команда

```
          MOV #SERV, @#250
```

всегда установит адрес ячейки SERV равным адресу обработки прерываний от блока управления памятью. Когда произойдет прерывание, в счетчике команд не окажется содержимого ячейки 250 общей шины. Действительно, при включенном блоке управления памятью виртуальный адрес 250 будет смещен, и в счетчик команд поступит содкржимое ячейки, соотвествующей преобразованному адресу.

Адрес, который передается програмой в качестве входной точки процедуры обработки прерываний, является виртуальным, так что он имеет отношение к блоку управления памятью. Предположим, что программа объявляет третью страницу нерезидентной, включается блок управлением памятью и затем выполняются команды:

```
          MOV #60000, @#250
          TST @#60000
```

Первая из них корректна. Хотя в ней фигурирует значение 600000, никаких ссылок на содержание ячейки с таким виртуальным адресом не делается и защита памяти не нужна. Поэтому команда благополучно установит 60000 в качестве входного адреса программы обработки прерываний.

Вторая же команда пытается обратиться к нерезидентной третьей странице. В результате блок управлением памятью игнорирует ее и передает управление по виртуальному адресу 250. Процессор загрузит содержимое виртуального адреса 600000 в счетчик команд для того, чтобы передать управление программе обработки прерываний. Это, однако, вновь приведет к срабатыванию механизма зашиты памяти; снова произойдет прерывание и т. д. В итоге программа зациклится.

Режим работы процессора

Привилегированных и непривилегированных пользователей в системе с разделенным

временем различают по режимам, в которых они могут работать с процессором. В системе имеется два режима: оперативный и обычный (пользовательский). Сразу после включения процессор перейдет в оперативный режим и останется в нем до тех пор, пока будет выполняться системная программа или программа привилегированного пользователя. Для непривилегированного пользователя операционная система изменит режим работы процессора, предварительно предприняв шаги к тому, чтобы пользователь не смог самовольно его изменить.

Наиболее характерное отличие заключается в том, что команда HALT является некорректной в режиме пользователя. Вместо останова процессора она приведет к прерыванию в ячейке 4 или 10. (В какой именно, зависит от процессора).

Режим задается 14-м и 15-м битами слова состояния процессора PS. Слово состояния имеет на общей шине адрес 77776. Если оба бита равны 0, то процессор работает в оперативном режиме. Если же это единицы, то режим обычный. На процессоре не допускается, чтобы значение битов было различным.

Когда блок управления памятью получает виртуальный адрес, то он прежде всего анализирует слово состояния, чтобы определить в каком режиме находится процессор. В зависимости от режима в описанном ранее алгортме вычисления смещения используется тот или иной набор регистров. Содержимое этих наборов в целом не зависит одно от другого. Твким образом, допустимо (это практикуется), что в пользовательском режиме программам будет закрыт доступ к регистрам ввода-вывода.

Тем не менее, выполняющейся в обычном режиме программе должны быть предоставлены возможности для осуществления ввода-вывода. Мы уже знаем, что это достигается при помощи системных подпрограмм, и сейчас можно более детально ознакомиться с тем, что происходит. Давайте в последний раз напишем программу печати на терминале буквы B. Мы достигнем цели командой TRAP в обычном режиме. Считая, что перед входом в программу процессор находится в оперативном режиме, установим программное прерывание:

```
        MOV #TRP, @#34
        MOV #340, @#36
```

Метку TRP мы выбрали в качестве входной метки процедуры обработки программных прерываний и установили приоритет последней равным 7, так что можно не беспокоится по поводу прерываний от внешних устройств. Заметьте, что значения адресов программных и внешний препываний определены в виртуальном адресном пространстве оперативного режима. Они будут доступны в другом режиме, если только какая-то страница виртуального адресного пространства обычного режима соотвествует тем же самым физическим ячейкам. Обычно же блок управления памятью накладывает запрет на доступ к этим векторам в пользовательском режиме.

В ячейке с меткой TRP имеем:

```
   TRP:  MOVB #102, @#TRB
         RTI
```

Естественно, что идентификатор TRB должен быть где-то описан. Предполагаем, что регистр R7 настраивается, как обычно, полагая TRB=177566.

Метке TRP загрузчик поставит в соответствие некий виртуальный адрес. При помощи блока управления памятью можно убедиться в том, что соотвествующая ему ячейка общей шины недоступна программам в обычном режиме В крайнем случае ее можно установить только на чтение. Тогда программа обычного режима могла бы выполнить команду JMP TRP, не вызвав обращения к блоку защиты памяти. Все же, поскольку регистры устройства ввода-вывода недоступны программам обычных пользователей, команда с меткой TRP, по видимому, не выполняется.

Связь между адресными пространствами

Системная программа EMT может выбрать параметр из младшего байта команды EMT. Предположим теперь, что обращение к команде EMT происходит, когда процессор находится в обычном режиме. Программа EMT сохраняет значение регистров R0 до R5 в стеке, и поэтому адресом возврата оказывается 14(SP). Он загружается в R0:

```
        MOV 14(SP), R0
```

Теперь сама команда EMT расположена по виртуальному адресу -2(R0), но в виртуальном пространстве обчного режима. Программа EMT, однако, будет функционировать в оперативном режиме. Допустим, что она была загружена с виртуального адреса 2000 обычного пространства. Если она заканчивается командой

```
        MOV --(R0), --(SP)
```

то в стек запишется содержимое виртуального адреса 2000 пространства оперативного режима. Это будет та же самая физическая ячейка памяти, если только нулевая страница в обоих режимах одинаково распределена.

Вместо предыдущего способа в системном вызове нужно использовать команду пересылки из пространства предыдущей команды MFPI (Move From Previous Istruction space). Это одноадресная команда. Ее операнд интерпретируется как результат вычисления исполнительного адреса, соотвествующий виртуальному адресу предшествующего пространства (в нашем случае пользовательского). Команда заносит содержимое этого адреса в системный стек. Поэтому программа EMT должна заканчиваться так:

```
        MFPI --(R0)
```

Обратная связь — от стека к адресу предшествующего пространства — достигается аналогичным способом при помощи команды записи в пространство предшествующей команды MTPI (Move To Previous Istruction space).

Предшествующий режим определяется как режим, в котором процессор находится перед последним программным или внешним прерыванием. Причем когда одно из них происходит, в слово состояния процессора заносятся значения битов второго слова вектора прерывания, за исключением битов 12 и 13: их состояние определяется предшествующими значениями битов 14 и 15 в PS. В командах MFPI и MTPI осуществляется проверка 12-го и 13-го битоа PS, позволяющих определить, о каком виртуальном пространстве идет речь.

Системные стеки

В каждом из режимов процессор заводит особый указатель стека. Указатели представляют собой совершенно различные регистры процессора, которые устанавливаются независимо один от другого. В то же время сам по себе не может предпочесть один стек другому: выбор предопределен режимом, в котором находится процессор.

Поэтому, если в программе определено выражение SP=6%, то любая ссылка на идентификатор SP будет интерпретироваться как обращение к стеку оперативного режима, если процессор находится в оперативном режиме, и как обращение к стеку обычного режима, если процессор находится в этом режиме.

Система команд процессора БК

Команды процессора К 1801 ВМ1 условно можно разделить на 4 группы: одноперандные команды, двухоперандные команды, команды передачи управления и безоперандные команды.

Двоичный код безоперанной команды содержит только код операции — информацию для процессора о том, что нужно делать по этой команде.

Двоичный код одноперандной команды содержит код операции и информацию для

процессора о местонахождении обрабатываемого числа (операнда), над которым нужно произвести операцию.

Двухоперандные команды, помимо кода операции, содержат информацию для процессора о местонахождении 2-х чисел (операндов). Напимер, для сложения двух чисел команда должна содержать код операции сложения и информацию о том, откуда взять слагаемые.

Далее коды команд процессора, а также коды операндов будем писать в восьмеричной системе счисления (как и коды адресов). Однако полезно помнить, что команды и операнды хранятся в памяти в двоичном коде, восьмеричный код мы будем использовать только для удобства.

```
       0  101 000 110 001 111  --- двоичный код
      ␣   ␣   ␣   ␣   ␣   ␣    --- восьмеричный код
```

На рисунке показано, как переводить двоичный код числа в восьмеричный. Число 050617 в восьмеричной системе счисления получено из 16-разрядного двоичного кода 0101000110001111 таким образом. Начиная с младшего разряда (справа) двоичное число делится на триады (группы по 3 цифры). Правда, старший разряд 16-разрядного числа при этом остается без “соседей”. Затем для каждой триады записывается ее представление в восьмеричной системе счисления. В результате вместо 16-разрядного кода числа получаем 6-разрядный восьмеричный код. Разумеется, при написании программ удобнее работать с 6-разрядными кодами команд и чисел, чем с 16-разрядными — благо, все имеющиеся программы, обепечивающие программирование в кодах, включая пультовый отладчик, понимают восьмеричную систему счисления.

Способы адресации операндов

В одноперндных командах процессора первые 4 восьмеричные цифры определяют код операции. Оставшиеся 2 цифры в коде команды процессор использует для определения местонахождения операнда, над которым нужно произвести операцию. Например, команда 005004 обнуляет регистр R4. Здесь первые 4 цифры кода команды “0050” являются кодом операции и указывают на то, что процессор должен произвести обнуление. Оставшиеся 2 цифры “04” указывают, что происходит обнуление регистра R4. Нетрудно догадаться, что последняя цифра является номером того регистра, содержимое которого используется процессором при определении местонахождения операнда. Предпоследняя цифра (код способа адресации) указывает, каким образом содержимое регистра используется при определении местонахождения операнда, то есть определяет способ адресации.

Регистровый режим адресации

В режиме ячейкой является сам указанный регистр. Под прямой адресацией мы подразумеваем, что поле операнда показывает действительный адрес и никаких вычислений не происходит.

```
         INC R3
```

Косвенный режим адресации

В косвеном режиме ячейкой является восьмибитовый байт или слово, адрес которого находится в указанном регистре.

```
         CLR @R5
```

Режим автоувеличения

Сразу после доступа к регистру для получения адреса, он автоматически увеличивается на 1, если операция байтовая, и на 2 если операция - словная.

```
        CMP (R3)+, R3
```

Косвенный режим автоувеличения

Число в указанном регистре используется, как адрес некоторой ячейки памяти; содержимое этой ячейки берется затем как адрес ячейки. Хотя содержимое указанного регистра увеличивается сразу же после его использования для получения адреса, это увеличение происходит на 2 и никогда на 1, даже если инструкция байтового типа.

```
        @(R1)+
```

Режим автоуменьшения

Уменьшение содержимого регистра происходит прежде его использования в качестве адреса. Для байтовых операций происходит уменьшение на 1, для словных инструкций — на 2.

1. -(R0)

Индексный режим

Получив содержимое регистра, процессор прибавляет к нему число, называемое индексом. Но содержимое регистра при таком сложении не изменяется. Полученное в результате сложения число используется как адрес ячейки памяти.

```
        X (R1)
```

Косвенный индексный режим

@ — указывает, что используется косвенный режми. Адрес ячейки вычисляется так: берется содержимое регистра, к нему прибавляется индекс, и результат используется как адрес ячейки памяти, содержимое которой является адресом, над которым выполняется операция. Где X — некоторое число.

```
        @X(R0)
```

Одноопернадные команды

Каждая команда, помимо своего числового кода, имеет свою мнемонику — обозначение в виде последовательности латинских букв, что используется для программирования на языке ассемблера. В случае обработки байтовой операцией обозначается - "B".

- CLR — очистка регистра.
- COM — по этой команде образуется инверсный код.
- INC — инкремент, увеличение содержимого на 1.
- DEC — декремент, уменьшение содержимого на 1.
- ADC — увеличивает значение операнда на содержимое разряда.
- SBC — уменьшает значение операнда на содержимое разряда.
- TST — тестирование значения операнда.
- ROR — циклический сдвиг значений разрядов вправо.
- ASL — тоже самое, но влево.
- SWAP — меняет местами старший и младший байты.
- MFPS — пересылает слово состояния процессора в место, определяемое полем адресации операнда.

Двухоперандые команды

Код двухоперндной команды, кроме кода операции, должен содержать 2 поля адресации

команды. Певое поле адресации операнда, обозначаемое в коде команды двумя буквами “SS”, определяет метонахождение 1-го операнда команды и называется полем адресации операнда источника. Второе поле адресации операнда обозначается в коде команды буквами “DD”, определяет местонахождение 2-го операнда команды и называется полем адресации операнда приемника. Первый операнд команды называется операндом источника, второй операнд — операндом приемника.

- MOV — пересылка по адресу, определяемому полем адресации.
- CMP — данная команда вычитает из операнда источника операнд приемника. Но при этом значения опрерандов не изменяется.
- BIT — значение каждого разряда результата определяется логическим умножением значений соотвествующего разряда операнда источника и операнда приемника.
- BIC — по этой команде сбрасываются в “0” разряды операнда.
- ADD — команда суммирует операнд источника с операндом приемника.

Переходы

- BR — переход.
- BNE — переход, если разряд Z слова состояния процессора сброшен в “0”.
- BNQ — переход в этой команде произойдет, если разряд Z слова состояния процессора установлен в “1”.
- BPL — переход произойдет, если разряд N слова состояния процессора сброшен в “0”.
- BMI — переход произойдет, если к моменту исполнения команды разряд N слова состояния процессора установлен в “1”.
- BVC — переход произойдет, если разряд V слова состояния процессора установлен в “1”.
- BGE — по команде произойдет переход, если содержимое регистра R1 оказалось больше или равно содержимому R2.

```
        MOV R1, R2
        CMP R1, R2
        BGE M1
        MOV R2, R3
M1:     . . . . .
```

- BLT — если вместо команды BGE поставить команду BLT, то переход по команде BLT произойдет, если содержимое R1 окажется меньше содержимого R2, в результате программа получит в R3 минимальное значение.

```
        MOV R2, R2
        SMP R1, R2
        BLT M1
        MOV R2, R3
M1:     . . . . .
```

- BLE — если команда BLE следует за командой сравнения двух операндов, то переход произойдет, если операнд источника меньше или равен операнду приемника.
- SOB — вычесть единицу и сделать переход, если не 0. Значение смещения занимает 6 младших разрядов кода команды и расматривается как число без знака — так как переход по этой команде происходит только в обратном направлении (в сторону уменьшения адресов). Для получения кода команды со смещением, отличным от “0”, к коду команды прибавляется значение смещения, равное значению выражения:

```
адрес команды + 2 - адрес перехода/2
```

Текст программы на ассемблере:

```
        MOV #1777, R0       ; длительность звука
        MOV #400, R1        ; длительность полупериода
M3:     MOV #100, @#177716  ; первый полупериод
        MOV R1, R2
M1:     NOP
        MOV #0, @#177716    ; второй полупериод
        MOV R1, R2
```

```
M2:     NOP
        SOB R2, M2
        SOB R0, M3
```

- JMP — команду можно использовать для перехода на любой адрес программы. Поле адресации в коде команды задает не адрес операнда, а адрес, с которого будет

продолжено выполнение программы после исполнения команды JMP. Поэтому в команде JMP недопустимо использование прямого регистрового способа адресации, так как передача управления на регистр процессора не имеет смысла.
- JSR — по команде адрес возврата запоминается в регистре, номер которого указывается в коде команды вместо буквы R, а содержимое самого регистра процессора до этого запоминается в стеке. Если в коде команды указан номер регистра R7, то адрес возврата запоминается в стеке.
- RTS — команда возвращает управление на адрес возврата, который по команде JSR был записан в регистр процессора. Номер этого регистра должен быть указан в коде данной команды вместо буквы "R".

Пример на языке ассемблера:

```
        JSR PC, @#7500
        . . . . .
        MOV #14, R0
        EMT 16          ; вывод на эран
        RTS PC
```

Цифровые базовые матричные кристалы типа К 1801 ВП1

Повышение степени интеграции прежде всего связано с применением метал-окисел-полупроводник транзисторов. Одним из первых больших микросхем нп основе n-каналальных структур был создан тип К 1801 ВП1, с проектными нормами длины канала 3 мкм. Его кристал, размером 4,2 * 4,2 мм условно разделен на внутреннюю и переферийную части с 43 контакными площадками.

Внутренняя часть большой интегральной микросхемы представляет собой матрицу 13*40 из 520 кристаллов типа А. Эта матрица имеет дополнительно два ряда по 40 матричных базовых ячеек типа В для реализации усилительных функций внутри матрицы. Наборы элементов, входящих в матричную базовую ячейку типа А, и усилительных типа В.

Каждая матричная базовая ячейка содержит 10 транзисторов и обеспечивает разветвление по выходу ячейки, равное 3. Усилительная матричная базовая ячейка содержит четыре транзистора, позволяющие расширить нагрузочную способность матричных базовых ячеек, и обспечивает коэффициент разветвления по выходу, равный 10. Периферийная часть большой интегральной микросхемы представляет собой ячейку, каждая из которых содержит 20 транзисторов, и контактную площадку, что позволяет осуществить 40 входов-выходов. Между контактными площадками питания и площадкой смещения размещен генератор смещения подложки. Шина земли выведена на контактную площадку. Библиотека фукциональных ячеек содержит 69 вариантов матричных фукциональных ячеек и 11 вариантов площадок функциональных ячеек.

Электрические параметры К 1801 ВП1

Параметр	Не более	Не менее
Напряжение питания Ucc	4,75 В	5,25 В
Выходное напряжение низкого уровня Uol, при Iol=4ма	-	0,4 В
Выходное напряжение высокого уровня Uoh, Ioh=1ма	2,7 В	-
Входное напряжение низкого уровня Ull	-	0,6 В
Входное напряжение высокого уровня Uih	2,4 В	-
Ток потребления Icc	-	180 мА
Ток утечки яо входу I	-	1 мкА
Среднее время задержки на линейном элементе при нагрузке на два входа, U=5В	-	5 нс
Максимальная входная частота fclc	-	8 МГц
Емкость входа-выхода Si/o	-	15 пФ

Корпус микросхемы — поликристалический.

Шины питания и земли включены в переменные слои металлизации и подлежат разводке. Разводка линий связи на кристале двухслойная, причем слой поликремния изменяемый, алюминия — переменный. Межслойные соединения осуществляются с помощью переменного слоя контактов. Основным критерием отимальности разводки линий связи можно считать их наименьшую длину и проведение их преимущественно в слое алюминия. Поликремниевые слои в рабочей зоне для удобства разводки линий связи через каждые две ячейки имеют разрыв. Если необходимо провести более длинную поликремниевую линию связи, то в местах разрыва ставят алюминиевые соединения.

Динамические параметры и ток потребления каждой конкретной большой интегральной схемы определяется в процессе разработки. Время задержки основных функциональных ячеек без учета топологических связей (для собственной емкости) указано в таблице выше.

Для обеспечения заданного быстродействия большой интегральной схемы необходимо рассчитать динамические параметры основных цепей схемы с учетом реальных физических процессов в кристале и реальной трассировки. Ориентировочно расчет проводится с учетом максимальных значений RC-линий связи между логическими элементами. Сопротивление и емкость линии связи расчитывают по формулам:

```
C = Сэ * (N + Csi + Sal)
```

```
R = сумма (Ro + Rт)
```

- Csi — емкость поликремневой линии связи.
- Cal — емкость алюминевой линии связи
- Сэ — емкость затвора линейного элемента
- Rт — 10 кОм сопротивление ключевого транзистора
- Ro — 50 кОм сопротивление нагрузки транзистора.

При расчете необходимо учитывать топологические особенности компоновки, так как задержка, вносимая поликремниевым слоем, может быть сравнима с задержкой линейного злемента.

В зависимости от вида аппаратуры и ее конкретного применения на основе большой интегральной микросхемы типа К 1801 ВП1 позволяет заменить до 60 микросхем малой и средней степени интеграции. Это обеспечивает уменьшение линейных размеров апаратуры в 4-16 раз.