Лекция 6-7

Архитектура 16-разрядных процессоров

6.1. Общая характеристика процессоров

Восьмиразрядные процессоры представляли собой класс устройств, в которых из-за ограниченного числа полупроводниковых элементов, размещаемых в кристалле, функции универсального процессора реализованы не в полной мере. Эти процессоры с небольшими изменениями используются до настоящего времени в качестве ядра однокристальных микро-ЭВМ, которые широко применяются устройствах управления в качестве миниатюрных и недорогих устройств с ограниченными функциональными возможностями.

Разработка 16-разрядных МП резко изменили ситуацию. Благодаря им МП достигли уровня центральных процессоров больших стационарных ЭВМ. Вначале были разработаны три основные типы 16-разрядных МП первого поколения, которые выпускались различными фирмами мира: MC 68000 (Motorola), 8086 (Intel) и Z8001 (Zilog).

6.1.1. Стандартная архитектура 16-разрядных МП первого поколения

В 16-разрядных МП длина слова равна 16 бит, адресуемое пространство памяти составляет 1-16 Мбайт, оперативный блок состоит из 16 регистров общего назначения. Шина данных является 16-разрядной, используется расширенный набор команд, включающий команды умножения, деления и др, отсутствующие в 8-разрядных МП. Структуры шин процессоров различных производителей приведена на рисунке 6.1.

Адреса обычно представляют собой 20-24-разрядные слова. Среди устройств этого класса имеются процессоры с раздельными шинами данных и адресов Motorola (MC68000) с большим количеством выводов (64) и процессоры с совмещенными шинами данных и адресов фирмы Intel и Zilog (8086 Z8001), с меньшим количеством выводов (40 и 48 соответственно).

Самая сложная архитектура и высокая стоимость в MC 68000. МП 8086 при более простой архитектуре обладает минимальной стоимостью. Особенностью этих МП является отсутствие специального регистра, называемого аккумулятором. В качестве двух операндов, поступающих в АЛУ, могут использоваться числа, хранящиеся в любых регистрах общего назначения (РОН). Результат операции, выполненной в АЛУ, заносится в

регистр или в основную память. При адресации указываются два адреса операндов. Обычно первый адрес указывается только в РОН, второй – как в РОН, так и в ОЗУ.

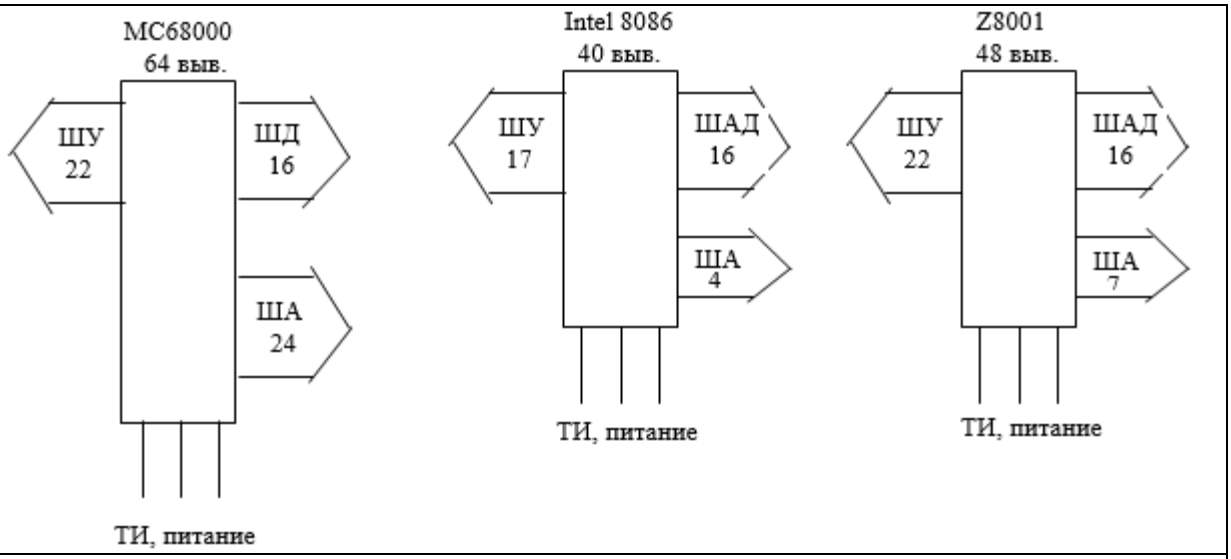

Рисунок 6.1 – Структура шин 16-разрядных процессоров

6.2. Адресное пространство 16-разрядных МП

Когда разрабатывались 8-разрядные МП, считалось, что для их практического применения достаточно 64 Кбайт пространства адресов. Поэтому в используемых в настоящее время 8-разрядных ЭВМ длина адресного слова составляет 16 бит. При такой длине адресного слова с адресами и данными можно обращаться одинаково, что значительно упрощает структуру ЦП. Однако со временем существенно увеличивается объем памяти на кристаллах, и как следствие этого программное обеспечение стало сложнее и разнообразнее. Кроме этого для построения универсальных компьютеров требуется значительно большее адресное пространство. В связи с этим 16-разрядные МП проектировались с длинным адресным словом: 20 – 8086, 23 – Z8001, 24 – MC68000.

6.2.1. Линейная и сегментная адресация

Если число разрядов адреса превышает длину информационного слова, возникают вопросы: в какой форме поместить такой адрес в регистр и каким способом осуществлять вычисление адреса? Для решения этих задач используют два способа адресации: *линейную* и *сегментную.*

При *линейной адресации* адрес представляет собой отдельное целочисленное значение. Вычисление адреса осуществляется с помощью операций сложения, приращения 16-ти разрядов слова двойной длины.В 16-разрядных МП выполняются аналогичные операции, т.е. адреса хранятся в 32-разрядных регистрах двойной длины. Архитектура такого процессора довольно сложная, но зато удобная для использования, т.к. все адресное пространство используется, как единое целое.

При *сегментной адресации* все пространство адресов делится на множество сегментов, т.е. пространство является сегментированным. Начальный адрес сегмента называют базовым. Порядок разбиения на сегменты может быть произвольным и после того, как он установлен, адрес можно представить с помощью номера (базового адреса) сегмента и смещения. Т.е. можно использовать два 16-разрядных регистра. Следовательно, максимальный размер одного сегмента составит 64Кбайт, а вычисления адреса сводится только к вычислению смещения. Для вычисления смещения можно воспользоваться тем же 16-разрядным АЛУ, что значительно упрощает структуру МП.

В МП MC68000 используется линейная адресация. 24-разрядный код адреса позволяет обращаться к 16 Мбайтам адресного пространства. Адреса хранятся в 32-разрядных регистрах. Операционный блок, кроме 16-го АЛУ, содержит два 16-разрядных сумматора, с помощью которых одновременно осуществляется вычисления 32-разрядных адресов. В МП 8086 и Z8001 применяется сегментная адресация. В 8086 используется базовый адрес сегмента и смещения, а в Z8001 – номер сегмента (7 старших бит) и 16 разрядов – смещение.

6.2.2. Размещение байтов и слов в памяти

Память в микро-ЭВМ на базе МП К1810 (8086) логически организована как одномерный массив *байтов*, каждый из которых имеет адрес в диапазоне *0000 – FFFFF*. Любые два смежных байта могут рассматриваться как 16-битовое слово. Младший байт имеет меньший адрес, старший – больший. *Адресом слова считается адрес его младшего байта.*

Полная информация, необходимая для определения его физического адреса содержится в адресном объекте *"сегмент-смещение*", который называется указателем адреса и содержит адрес сегмента и внутрисегментное смещение.

Для запоминания указателя адреса требуется *два слова памяти*, причем слово с *меньшим адресом* всегда содержит *смещение*, а слово с большим адресом – *базовый адрес сегмента* (рисунок 6.2).

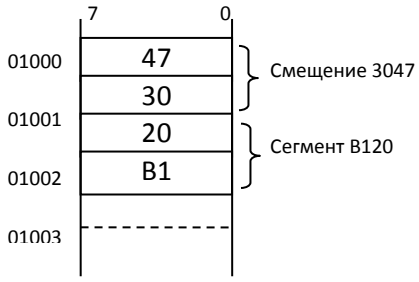

Рисунок 6.2 –Размещение слов

Команды, байты и слова данных можно свободно размещать по любому адресу, что позволяет экономить память, благодаря ее плотной упаковки. Однако, для экономии времени выполнения программ, целесообразно размещать слова данных в памяти по *четным адресам*, т.к. МП передает такие слова за один машинный цикл.

Слово с четным адресом называется *выровненным на границе слов*. Слова с нечетными адресами (не выровненные) также допустимы, но для их передачи требуется два цикла шины, что снижает производительность МП. Следует отметить, что интерфейсный блок процессора (BIU) инициирует необходимое для выборки слова число обращений к памяти автоматически, так что двукратное обращение к памяти не требует специального указания в программе. Особенно важно иметь выровненные слова для операций со стеком, так как в них участвуют только слова. Следовательно, указатель адреса стека SP необходимо *всегда* инициализировать на четный адрес.

Команды всегда выбираются словами по четным адресам, за исключением первой выборки после передачи управления по нечетному адресу, когда выбирается один байт. Поток команд разделяется на байты при заполнении конвейера команд внутри МП. Т.о. выравнивание команд не влияет на производительность и потому не используется.

5.2.3. Сегментация памяти и вычисление адресов

Пространство памяти емкостью 1 Мбайт представляется как набор сегментов, определяемых программным путем. Сегмент состоит из смежных ячеек памяти и является независимой и отдельно адресуемой единицей памяти емкостью 64 Кбайт. Каждому сегменту программой назначаются начальный (базовый) адрес, являющийся адресом первого байта сегмента в пространстве памяти. Начальные адреса четырех сегментов, выбранных в качестве текущих, записываются в сегментные регистры CS, DS, SS, ES. Для обращения к команде и данным, находящимся в других сегментах, необходимо изменить содержимое сегментных регистров, что позволяет использовать все пространство памяти емкостью 1 Мбайт. Частный случай загрузки всех

сегментных регистров нулями приводит к организации памяти характерной для МП580ВМ80, т.е фактически к отказу от сегментации памяти.

В сегментном регистре хранится 16 старших битов 20-разрядного начального адреса сегмента. Четыре младших бита принимаются равными нулю и дописываются справа к содержимому сегментного регистра при вычислении физических адресов ячеек памяти. Поэтому начальные адреса сегментов всегда кратны 16. Поскольку других ограничений на размещение сегментов в памяти нет, *сегменты могут быть соседними (смежными), неперекрывающимися, частично или полностью перекрывающимися*. Физическая ячейка памяти может принадлежать одному или нескольким сегментам.

Физический адрес сегмента памяти представляет 20-битовое число в диапазоне 00000-FFFFF, которое однозначно определяет положение каждого байта. *Логический адрес* ячейки памяти состоит из двух 16-битовых беззнаковых значений: *начального адреса сегмента*, который называют *базой или сегментом*, и *внутрисегментного смещения*, определяющего расстояние от начала сегмента до этой ячейки.

Для вычисления физического адреса база сегмента сдвигается на 4 бита влево и суммируется со смещением. Перенос из старшего бита, который может возникнуть при суммировании, игнорируется (рис.5.3).

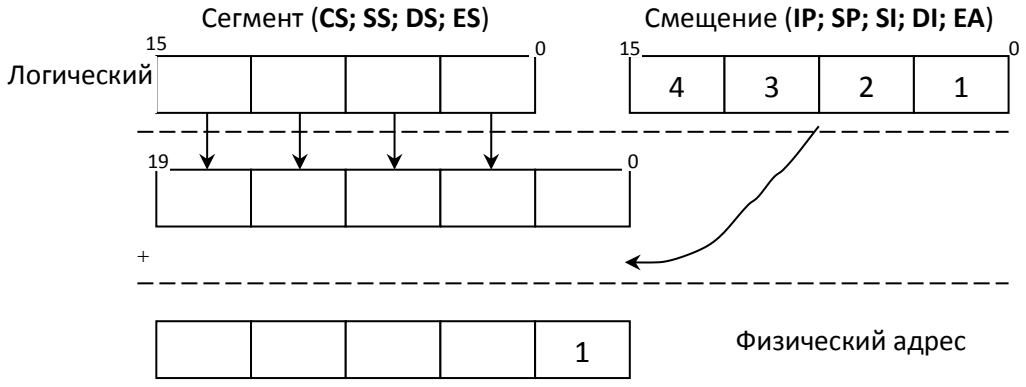

Рисунок 5.3 - Схема вычисления физического адреса

Это приводит к так называемой кольцевой организации памяти, при которой за ячейкой FFFFF следует ячейка с нулевым адресом. Аналогичную кольцевую организацию имеет и каждый сегмент.

Команды всегда выбираются из текущего сегмента кода в соответствии с логическим адресом CS:IP. Стековые команды всегда обращаются к текущему сегменту стека по адресу SS:SP. Если при вычислении эффективного адреса EA используется регистр BP, то обращение производится также к стековому сегменту, а ячейки стекового сегмента рассматриваются как ОЗУ с произвольной выборкой. Операнды, как правило, размещаются в текущем сегменте данных, и обращение к ним организуется по адресу DS:EA. Однако программист может заставить МП обратиться к переменной, находящейся в другом текущем сегменте.

По умолчанию предполагается, что цепочка-источник находится в текущем сегменте данных, а ее смещение задается регистром SI. Цепочка - *получатель* обязательно располагается в текущем дополнительном сегменте, а *смещение* берется из регистра DI. Команды обработки цепочки автоматически модифицируют содержимое индексных регистров SI и DI по мере продвижения по цепочке в направлении, задаваемом флагом DF. Источники логического адреса для различных типов обращения к памяти приведены в табл. 5.1.

Таблица 5.1 – Источники логического адреса

Тип обращения к памяти	Сегмент (умолчание)	Вариант	Смещение
Выборка команд	CS	нет	IP
Стековые операции	SS	нет	SP
Переменная	DS	CS, SS, ES	EA
Цепочка-источник	DS	CS, SS, ES	SI
Цепочка-приемник	ES	нет	DI
как базовый регистр	ES	CS, SS, DS	EA

Смена сегментного регистра осуществляется с помощью однобайтового префикса замены сегмента 001SR110, который ставится первым байтом команды. Двухбитовое поле SR содержит код сегментного регистра, используемого для вычисления физического адреса в данной команде: 00 – регистр EA; 01 – CS; 10 – SS; 11 – DS.

Сегментная структура памяти обеспечивает возможность создания позиционно независимых или динамически перемещаемых программ, что необходимо в мультипрограммной среде для эффективного использования оперативной памяти. Чтобы обеспечить позиционную независимость все смещения в программе должны задаваться относительно фиксированных значений, содержащихся в сегментных регистрах. Это позволяет произвольно перемещать программу в адресном пространстве памяти, *изменяя только содержимое сегментных регистров*.

5.3. Микропроцессорный комплект серии 1810

В микропроцессорном комплекте (МПК) серии 1810 (отечественный аналог процессора Intel 8086) реализуются и находят дальнейшее развитие идеи построения МП-систем на базе БИС с фиксированным набором команд. Микро ЭВМ, построенные на этом комплекте, могут работать с тактовой частотой 5 МГц и имеют 20-разрядную ША и 16-разрядную ШД, шины управления и арбитража. Шина арбитража служит для разделения времени доступа каждого процессора к магистралям микро-ЭВМ многопроцессорной системы. Основной схемой в МПК является БИС К1810ВМ86. Ее структура и режимы работы определяют основные особенности работы МП систем, построенных на МПК серии 1810. Кроме этого в состав комплекта входит генератор тактовых импульсов, буферные регистры, системный контроллер, программируемый таймер и др.

5.3.1. Микропроцессорная БИС К1810ВМ86

БИС представляет собой однокристальный 16-разрядный МП с мультиплексной 20-разрядной ША и 16-разрядной ШД и рассчитана на работу как в одно, так и в многопроцессорных системах. Процессор К1810ВМ86 на языке ассемблера совместим с МП КР580ВМ80, регистры и систему команд которого можно рассматривать как подсистему регистров и команд К1810ВМ86.

Эффективность работы МП 1810 существенно повышена за счет введения команд математических операций (включающих *умножение* и *деление*) над 8- и 16-разрядными числами, команд побитовой обработки чисел, команд *работы с массивами данных*, расширения видов прерываний работы МП, а также реализации *конвейерного* типа выполнения команд в самой БИС. МП может работать с памятью объемом до 1 Мбайт, обмениваться информацией с 64 Кбайт внешних устройств, имеет 256 типов различных прерываний.

5.3.2. Архитектура процессора

Обычно процесс выполнения команд включает этапы:

1. Извлечение КОП (операндов команды) из памяти.
2. Выполнение команды.
3. Запись результатов.

Как правило, в МП 580ВМ80 эти этапы выполняются последовательно, что приводит к недоиспользованию по временной загрузке шин микро-ЭВМ. В МП К1810ВМ86 процесс выполнения команд состоит из тех же этапов, однако, производится в двух раздельных процессорных блоках: в блоке *выполнения команд* (EU – *execution unit*) и *блоке сопряжения с магистралями* (BIU – *Bus interface unit*).

В функции BIU входит извлечение из памяти кода команд и их операндов, а также запись результата в память. Блоки могут работать независимо друг от друга, и, следовательно, процессы преобразования и передачи информации в них могут идти параллельно.

На рис.5.4, для сравнения, приведены два варианта различных способов выполнения команд одной программы в процессорах с последовательной (а) и параллельной (б) обработкой команд.

Рисунок 5.4 - Процесс выполнения команд в МП с последовательным (а) и параллельным (б) в блоках EU и BIO

Предполагается, что на первом этапе оба процессора выполняют первую команду. *Первая команда* состоит из двух этапов: Выполнение 1 и запись результата в память (запись 1). *Вторая команда* – получение кода (Код 2), выполнение 2. *Третья команда* – получение кода (Код 3), получение операнда (Опер. 3), выполнение 3. *Четвертая команда* – получение кода 4, выполнение 4. Как видно из рис.5.4 в МП К1810ВМ86 за одно и то же время количество выполняемых и полученных команд больше, при этом шины микро-ЭВМ используются для обмена более эффективно.

Упрощенная структурная схема БИС К1810ВМ86 приведена на рис.5.5. Блок сопряжения с шинами BIU производит все пересылки данных и кодов для EU. Пересылки между памятью или внешними устройствами осуществляется по требованию EU. В то время как EU занят выполнением команды, блок BIU получает последующие в программе коды из памяти. *Блок выполнения команд* EU имеет 16-разрядные АЛУ с регистром состояния и флажками управления, а также РОНы. Все регистры и внутренние магистрали блоки 16-разрядные. Блок не имеет связи с внешними шинами МП.

На АЛУ поступают коды команд из конвейера команд, расположенного в BIU. Если в результате дешифрации кода команд в АЛУ необходимо получение одного или нескольких операндов по внешним магистралям МП, то EU запрашивает BIU на получение и размещение необходимых данных в BIU.

Несмотря на то, что все адреса, с которыми оперирует EU, 16-разрядные, BIU производит необходимые преобразования адресов так, чтобы EU имел возможность обращаться по всему возможному адресному пространству (1 Мбайт) МПС. BIU считывает команды с памяти и сохраняет их в конвейере команд, где может быть размещено до 6 инструкций. Это позволяет BIU выдавать их в EU по мере надобности без дополнительной загрузки внешней шины. BIU организует получение нового кода команды как только два байта из конвейера будут переданы в EU.

В большинстве случаев в BIU находится хотя бы одна команда и EU не простаивает, пока очередная команда будет извлечена из памяти. Коды подаются в EU последовательно, так как они записаны в программе.

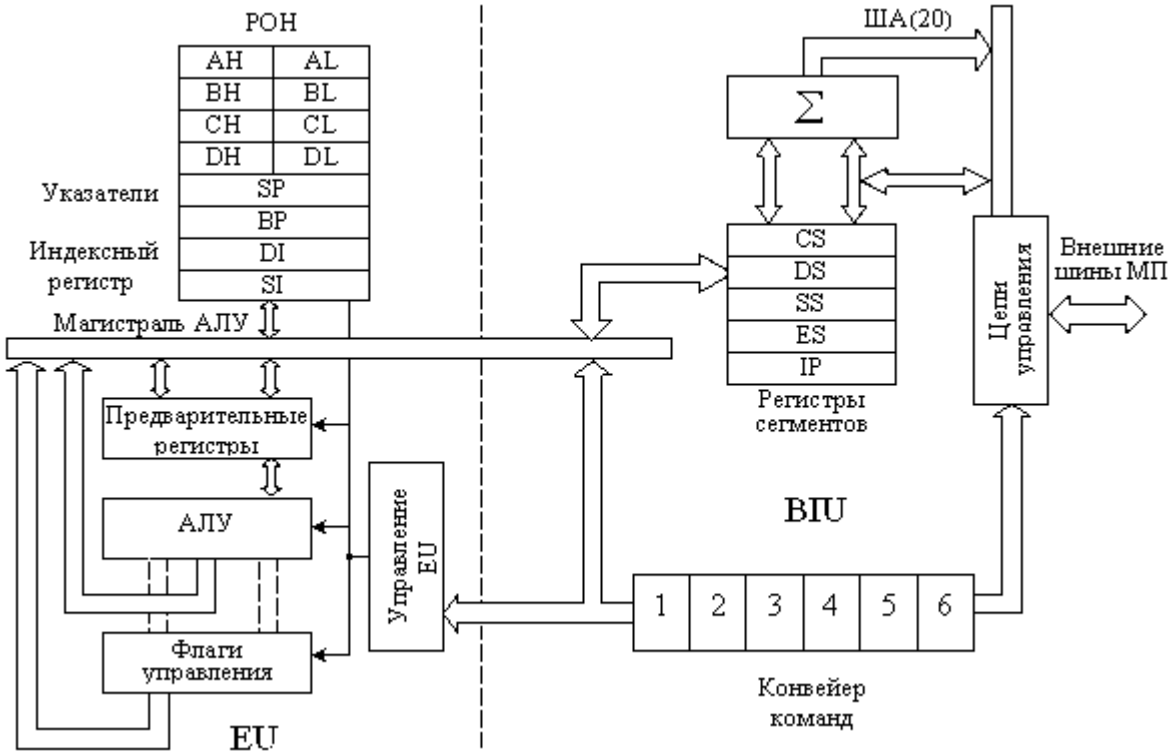

Рисунок 5.5 – Схема шестнадцатиразрядного микропроцессора К1810ВМ86

Если EU выполняет команду передачи управления в другое место программы, то BIU очищает конвейер команд, получает код из нового адреса, передает его в EU и начинает заполнять конвейер заново. Если EU требует обращения к внешнему устройству, то BIU приостанавливает процесс получения команд в конвейер и организует необходимый цикл обмена данными.

5.3.3. Адресация памяти компьютера

Любая ячейка памяти МП имеет два типа адресов: физический и логический. Физический адрес представляется 20-разрядным числом и однозначно определяет любую из 1 Мбайт ячеек памяти. В 16-разрядной системе адреса расположены в диапазоне от 0 до FFFFF. Весь обмен информацией МП с памятью осуществляется с использованием физических адресов. Программы же больше используют логические или физические адреса и позволяют записывать команду без предварительных знаний места, где эта команда будет размещена в памяти. Это дает программисту возможность маневрировать при распределении памяти, увеличивает гибкость программного обеспечения.

Логический адрес состоит из двух основных частей: значения *базы сегмента* и значения *смещения в сегменте*. Базовый адрес и смещение в сегменте отображаются 16-разрядными числами. Как только BIU обращается к памяти, базовый адрес формирует физический по принципу: Значение базы сегмента смещается на четыре разряда влево, и полученное 20-разрядное число (с четырьмя нулями в младших четырех разрядах) складываются со значением смещения в сегменте. Таким образом, база сегмента (с четырьмя нулями, добавленными в качестве младших разрядов) задает для памяти сегменты длинной 64 Кбайт, а значение сегмента в смещении – расстояние от начала сегмента до искомого адреса памяти. Максимально возможное смещение в сегменте равно 64 Кбайт. В любой момент времени программа может осуществлять доступ к одному из четырех сегментов:

1) сегменту текущего кода (*Current Code Segment* – CS);
2) сегменту текущих данных (*Current Date Segment* - DS);
3) сегмент текущего значения стека (*Current Stack Segment* – ES);
4) дополнительный сегмент текущих значений (*Current Extra Segment* – ES).

Каждый из этих сегментов может быть задан с помощью записи числа в соответствующий 16-разрядный регистр сегментов МП БИС. В большинстве случаев в МПК К1810 сегменты могут быть заданы в программе произвольно по усмотрению программиста.

В зависимости от команд, блок BIU получает информацию о логическом адресе памяти из различных регистров МП. *Коды команд* всегда извлекаются из адреса памяти, определяемого *содержимым регистра текущего кода* и регистра *указателя команд* IP (*Instruction Pointer*). В IP записано смещение в сегменте текущего кода. *Команды со стеком* всегда используют для адресации SS и регистр – *указатель стека* SP (*Stack Pointer*), где записано смещения в сегменте текущего значения стека. *Данные или переменные* в командах извлекаются из памяти, адреса которых расположены в *сегменте текущих данных*, хотя по директивам команд это можно сделать и с помощью всех остальных сегментов. При вычислении физического адреса смещение в сегменте задается в EU, так как в нем происходит дешифрация кода команд и определение сегмента памяти, с которых будет работать БИС в текущий момент времени.

5.3.4. Регистры МП К1810ВМ86

МП содержит три группы регистров. К *первой группе* относятся РОН, используемые для хранения промежуточных результатов. Ко *второй* группе относятся *указатели и индексные регистры*, предназначенные для размещения или извлечения данных из выбранного сегмента памяти.

Содержание этих регистров определяет значение смещения в сегменте при задании логического адреса. К *третьей* группе относятся *регистры сегментов*, задающие начальные адреса (базы) самих сегментов памяти.

МП имеет регистр флаговых разрядов, используемых для указания состояния АЛУ при выполнении различных команд и управления его работой.

Таким образом в МП располагается тринадцать 16-разрядных регистров и девять флаговых разрядов АЛУ. Последний, тринадцатый регистр, называется указателем команд *IP (Instruction Pointer*), выполняет функции, аналогичные функциям программного счетчика в МП КР580ИК80 и не является программно доступным регистром. Доступ к его содержимому может быть осуществлен с помощью команд передачи управления.

Группа РОН включает в себя семь 8-разрядных регистров МП КР580ИК80 и один добавленный 8-разрядный регистр для того, чтобы объединить все регистры в четыре 16-разрядные пары (Рис.5.6). К ним может быть организован программный доступ как к 8- или 16-разрядным регистрам. 16-разрядные регистры обозначаются AX, BX, CX и т.д. При обращении к ним, как к 8-разрядным регистрам, их обозначают AL, AH, BL, BH и т.д. Все эти регистры могут быть использованы при выполнении арифметических и логических команд. Однако, есть команды, использующие определенные регистры для специфических целей, тогда применяют мнемонические обозначения: аккумулятор (*accumulator*), база (*base*), счет (*count*), данные (*data*).

Группа указателей и индексных регистров состоит из четырех 16-разрядных регистров.

SP – Stack Pointer } Регистры указатели

BP – Base Pointer

SI – Source index } Индексные регистры

DI – Destination index

Обычно эти регистры содержат информацию о смещении по адресам в выбранном сегменте и позволяют компактно писать программы каждый раз непосредственно, не приводя используемого адреса. С их помощью производится вычисление адресов программ. Чаще всего в регистрах *указателях* записано адресное смещение по отношению к *стековому* сегменту, а в *индексных* регистрах – адресное смещение по отношению к *сегменту данных*.

Группа регистров сегментов состоит из четырех 16-разрядных регистров:

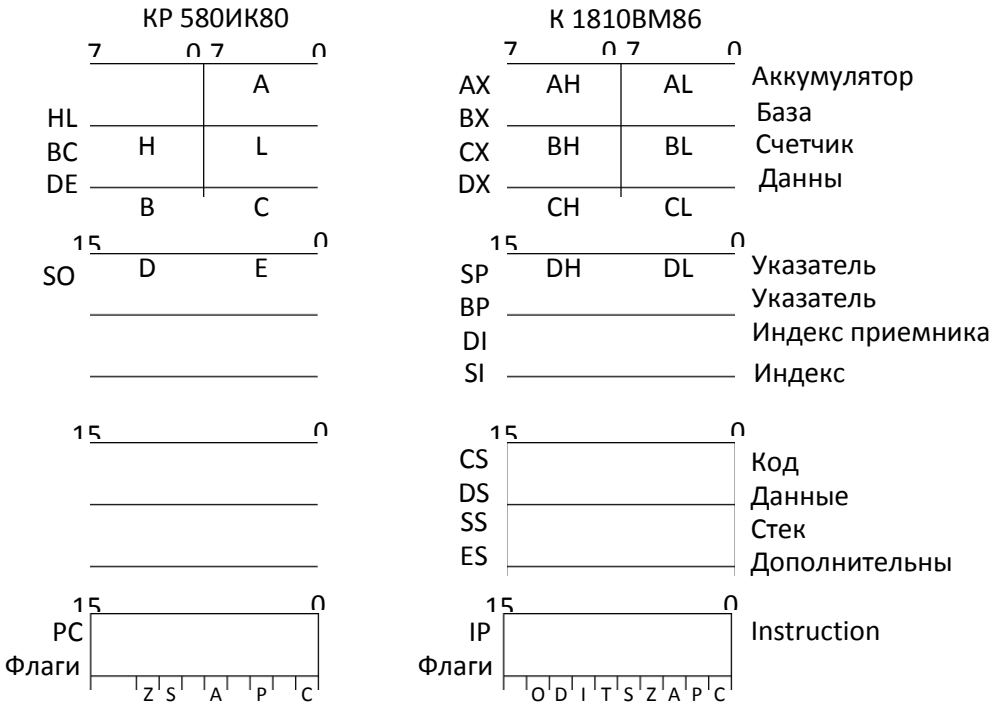

Рисунок 5.6 - Регистры МП серий 580 и 1810

Извлечение кодов команд производится из сегментов текущего кода с использованием адресного смещения, задаваемого указателем команд IP.

Сегмент для извлечения операндов команды обычно указывается путем записи специального однобайтового префикса перед командой. Префикс имеет специальную кодировку, позволяющую МП отличить его от кода команды.

Регистр состояния (Флаговый регистр) содержит шестнадцать триггеров, из которых используется только 9. Эти триггеры отображают состояние процессора при выполнении последней арифметической или логической команды. Пять триггеров аналогичны МП 580-й серии. Дополнительные триггеры сигнализируют:

OVERFLOW – переполнение (при выполнении операций с числами МП имеет дополнительные флаги управления

Флаги
- DIRECTION – направление – указывает направление работы с массивами (автоматическое увеличение или уменьшение на единицу адреса массива)
- INTERRUPT – прерывание – определяет для МП возможность реагирования на прерывание.

TRAP – (западня, ловушка) устанавливает для МП пошаговый режим.

Кроме основных функций, соответствующих названию регистров, общие регистры выполняют специальные функции, указанные в таблице.

Регистр	Название	Специальная функция регистра
AX	*Аккумулятор*	*Умножение, деление, ввод-вывод слов*
AL	Аккумулятор - младший байт	Умножение, деление, ввод-вывод байтов, преобразование байтов, десятичная арифметика
AH	Аккумулятор - старший байт	Умножение и деление слов
BX	Базовый регистр	Адресация по базе; преобразование адресов
CX	Счетчик	Подсчет циклов; подсчет элементов цепочек
CL	Счетчик (младший байт)	Реализация параметрических сдвигов
DX	Регистр данных	Умножение и деление слов; косвенный ввод-вывод
SP	Указатель стека	Операции с использованием стека
BP	Указатель базы	Базовый регистр
SI	Индекс источника	Указатель цепочки-источника, индекс, регистр
DI	Индекс приемника	Указатель цепочки-приемника, индексный регистр

5.3.5. Задание режима работы МП К1810ВМ86

Для задания режима работы используется вывод $\overline{MN/MX}$. Подача на этот вход напряжения +5 В указывает на то, что в МПС имеется лишь одна БИС МП (минимальная конфигурация системы). При этом МП вырабатывает на своих электродах все необходимые управляющие сигналы для связи с памятью и периферийными устройствами. При максимальной конфигурации

МПС $\overline{MN/MX}$ =0. МП выдает дополнительно на три вывода, подсоединяемые к контроллеру магистралей, управляющие сигналы.

$\overline{LOCK}$ - (запрет на пользование магистралями) – информирует устройства системы, что они не должны пытаться запрашивать шину.

$\overline{RQ}/\overline{GT0}$ $\overline{RQ/E0}$ } Запрос и получение подтверждения от основной МП БИС на возможность доступа к магистралям.
$\overline{RQ}/\overline{GT1}$ $\overline{RQ/E1}$ -

При работе максимальной конфигурации МП выдает восемь сигналов состояния процессора (ST0-ST7), которые могут использоваться внешними устройствами.

ST2	ST1	ST0	- указывают тип машинного цикла
0	0	0	Обслуживание прерываний
0	0	1	Чтение внешних устройств
0	1	0	Запись во внешние устройства
0	1	1	Останов
1	0	0	Извлечение кода команд
1	0	1	Чтение памяти
1	1	0	Запись в память
1	1	1	Не используется

Выводы ST3 и ST4 указывают на тип регистра сегментов, используемых при вычислении физического адреса.

0	0	ES
0	1	SS
1	0	CS (или никакой)
1	1	DS

Выводы QS0 QS1 Выполняется текущая операция с конвейером команд.

0	0	Нет операции
0	1	Извлекается первый байт команды из конвейера
1	0	Конвейер очищается
1	1	Очередной байт извлекается из конвейера

WAIT – ожидание

HLD – прямой доступ в память (запрос)

5.4. Система команд 16-разрядного процессора 8086

Система команд ВМ86 содержит 91 мнемокоманду и позволяет совершать операции над байтами, двухбайтовыми символами, отдельными битами, а также цепочками байтов и слов.

По функциональному признаку система команд ВМ86 разбивается на 6 групп:

1) Пересылка данных;
2) Арифметические операции;
3) Логические операции и сдвиги;
4) Передача управления;
5) Обработка цепочек;
6) Управление процессором.

5.4.1. Команды пересылки данных

Команды пересылки данных составляют 4 подгруппы:

1) общие;
2) обращение к стеку;
3) ввода-вывода;
4) пересылка цепочек.

Эти команды, за исключением POPF и SAHF, не влияют на флаги. Команда MOV осуществляет пересылку содержимого источника src в получатель dst.

Команда **MOV r_1/m, r_2/m** обеспечивает пересылки регистр-регистр/память-память при использовании любого общего регистра и любого способа адресации.

Команда **MOV r/m, d** позволяет передать непосредственные данные в общий регистр или ячейку памяти.

MOV sr, r/m и **MOV r/m, sr** осуществляют пересылки между сегментным регистром и регистром или памятью. При этом передаются только слова, а ячейка памяти может быть определена с помощью любого допустимого способа адресации.

MOV *a*c, m и **MOV m, *a*c** предназначены для загрузки и запоминания содержимого аккумуляторов AL и AX при использовании прямой адресации.

Обращение производится к текущему сегменту данных, и адрес, указанный в команде, представляет смещение в этом сегменте. Если пересылаются два байта, то младший располагается по указанному адресу, а старший – по следующему. Так как не существует непосредственной загрузки сегментных регистров, то команда MOV sr, r/m используется для

инициализации регистров SS, DS и ES, т.е. для определения соответствующих сегментов памяти. Если, например, в регистр DS необходимо загрузить число 8000, то используют две команды:

MOV AX, 8000H

MOV DS, AX

При этом в качестве промежуточного регистра обычно используют AX, т.к. команда MOV*ac*, d короче более общей команды MOV m/r, d.

Возможна также инициализация сегментных регистров из программной памяти при использовании префикса замены сегмента для замены DS на CS при вычислении адреса EA в следующих командах с прямой адресацией. При замене сегмента новый сегментный регистр указывается явно.

MOV DS, CS : ADS; Инициализация DS

MOV ES, CS : AES; инициализация ES

MOV SS, CS : ASS; инициализация SS

Следует помнить, что в МП BM86 обеспечивается защита процесса инициализации регистров SS и SP, состоящего из двух команд, от прерываний, чтобы гарантировать правильную работу стека.

MOV SS, CS : ASS

MOV SP, CS : ASP

Операнды ADS, AES, ASS, ASP обозначают адреса ячеек памяти, в которых хранятся базовые адреса соответствующих сегментов.

Команда

XCHG dst, sre; dst ↔sre

осуществляет обмен данными между источником и получателем и имеет два формата. Общий формат позволяет произвести обмен содержимым любой пары общих регистров, а также между общим регистром и ячейкой памяти при любом допустимом способе адресации. Указанный формат осуществляет обмен любого РОН и аккумулятора AX.

Команда XCHG AX,AX, код которой 90H, используется как команда операции NOP, обеспечивающая задержку на время 3T.

Однобайтная команда XLAT с кодом операции D7 предназначена для быстрого преобразования кодов и заменяет содержимое AL на байт из 256-байтовой таблицы, начальный базовый адрес которой содержится в регистре BX, т.е. содержимое AL используется как индекс таблицы, находящейся в сегменте данных и адресуемой регистром BX. При выполнении этой команды, к содержимому BX прибавляется содержимое AL, а полученный результат используется как смещение относительно DS. Адресуемый таким образом байт из памяти пересылается в AL.

Команды LEA, LDS, LES отличаются от других команд пересылки тем, что при их выполнении в адресуемый регистр (регистры) передаются не собственно данные из памяти, а адреса. Основное назначение этих команд –

инициализация регистров перед выполнением цепочечных команд, или перед вызовом подпрограммы.

LEA r,m обеспечивает вычисление эффективного адреса EA ячейки памяти в соответствии с указанным способом адресации и загрузку EA (а не содержимого адресуемой ячейки памяти) в указанный РОН. Такая операция может потребоваться, например, для загрузки начального адреса таблицы в регистр BX перед выполнением команды XLAT.

Стек, как обычно, организуется в ОЗУ и его положение определяется содержимым регистров SS и SP. Регистр SS хранит базовый адрес текущего сегмента стека, а регистр SP указывает на вершину стека в стековом сегменте. При каждом обращении к стеку пересылается одно слово, а содержимое SP модифицируется автоматически, при записи в стек оно уменьшается на 2, при чтении из стека – увеличивается на 2.

При всех достоинствах организации памяти ВМ86 она имеет некоторый недостаток, заключающийся в трудности манипуляции физическими адресами при необходимости их программной обработки.

5.4.2. Команды ввода-вывода

Ввод-вывод данных может осуществляться двумя способами: с использованием адресного пространства ввода-вывода и с использованием общего с памятью адресного пространства. При первом способе применяются команды IN, OUT, которые обеспечивают передачу данных между аккумуляторами AL или AX и адресуемыми портами.

При выполнении этих команд вырабатывается сигнал M/IO=0, который вместе с сигналами WR и RD позволяет сформировать системные сигналы IWO и IOR для управления операциями записи данных в порт или чтения из порта.

Команды IN и OUT могут использовать прямую адресацию (аналог с ВМ80) и косвенную, когда адрес порта содержится в регистре DX. В первом случае можно адресовать 256 портов. Во втором – до 64К 8-битовых или 32К 16-битовых портов. Косвенная адресация позволяет вычислять адреса портов при выполнении программы, и удобна при организации вычислительных циклов для обслуживания нескольких портов с помощью одной процедуры. Ячейки с адресами F8 – FF зарезервированы для системных целей, и использовать их в прикладных программах не рекомендуется.

При втором способе адресации обращение к портам не отличается от обращения к ячейкам памяти, т.е. для ввода-вывода можно использовать любую команду обращения к памяти при любом способе адресации. Однако следует учитывать, что команды обращения к памяти имеют больший формат и выполняются дольше, чем простые команды IN и OUT. Кроме того, усложняется декодирование 20-битового физического адреса порта и сокращается число адресов, которые могут использоваться для ячеек памяти.

МП может передавать по ШД байт или слово из ВУ. Чтобы слово передавалось за один цикл шины, адрес ВУ должен быть четным. Адрес

байтового ВУ может быть четным или нечетным, и соответственно порты этих ВУ должны подключаться к линиям младшего и старшего байта ШД. Для раздельного обращения к этим портам дешифрирование адресов осуществляется с учетом сигналов на линиях BHE и A0.

5.4.3. Цепочные команды (обработки строк)

Под цепочкой (строкой) понимают последовательность любых контекстно-связанных байт или слов, находящихся в смежных ячейках памяти. В системе команд МП К1810ВМ86 имеется пять однобайтных команд, предназначенных для обработки одного элемента цепочки за прием. Цепочной команде может предшествовать специальный однобайтовый префикс повторения REP. Число повторений задается регистром CX.

Например, последовательность команд

MOV CX, 500

REP MOVS DST, SRC

заставит МП выполнять команду MOVS 500 раз, уменьшая значение регистра CX после каждого повторения, до тех пор, пока <CX> не станет равным 0.

Остальные префиксы повторения, решение о продолжении или прекращении повторений принимают в зависимости от значения флага ZF:

REPE (repeat while equal) пока равно;

REPZ (repeat while zero) повторять пока нуль;

REPNE

REPNZ

Команда LODS *(load string)* пересылает операнд строка-источник, адресованный регистром SI, из сегмента данных в регистр AL или AX, а затем изменяет SI так, чтобы он указывал на следующий элемент строки. Его значение увеличивается, если флаг направления DF равен 0, и уменьшается, если DF=1.

Формат команды имеет вид

LODS SRC ; <AC>←<SRC>.

Допускается использование мнемокодов LODSB и LODSW, указывающих тип строки (цепочки).

Пример:

Сравнить строки DST и SRC длиной 500 байтов каждая. В случае обнаружения первого несовпадения, элемент строки LODS загрузить в регистр AL.

```
CLD
LEA   DI,  ES : DST
LEA   SI,  SRC
MOV CX, 5000
REPE CMPSB
  JCXZ  M1          ; Выйти, если CX=0, на метку M1
  DEC  SI           ;  Подправить регистр SI
  LODS  SRC         ; считать элемент в AL
  …….
M1:  …….
```

Команда сохранения строки STOS (stove string) служит для запоминания содержимого аккумулятора в элементе строки, адресуемом регистром DI. После выполнения команды содержимое DI увеличивается на 1. Формат команды

STOS DST ; DST:=<AC>

Сегментный адрес для этой команды всегда находится в сегменте ES, а префикс замены сегмента не используется. Тип элементов строки можно указывать с помощью мнемокодов STOSB и STOSW.

Пример: Просмотреть строку W_STRING длиной в 200 слов. Если обнаружен не нулевой элемент, то он и следующие за ним 5 слов обнуляются.

```
CLD
LEA   DI,  ES : W_STRING
MOV AX, 0
MOV CX, 2000
REPNE SCASW
JCXZ  ALLO
SUB    DI, 2
MOV  CX, 6
REP      STOS W_STRING
      …….
ALLO:  …….
```

Команда пересылки строк MOVS используется для копирования байта или слова из одной части памяти в другую. Она имеет формат

MOVS DST, SRC ; DST:=< SRC>

Здесь строка-источник находится в сегменте данных DS, а строка приемник – в дополнительном сегменте ES. Байт или слово источника,

адресуется регистром SI, а получатель DI. После выполнения команды значения SI и DI модифицируются для указания следующих элементов строк.

При флаге DF=0 осуществляется автоинкрементация, а при DF=1 – декрементация.

Пример: Скопировать 100 байтов из строки SRC в строку DST.

```
CLD                  ; Установить DF=0 для обработки строки слева направо
LEA   SI,  SRC       ; Занести смещение адреса SRC в SI, а
LEA   DI,  ES: DST ; смещение адреса DST в DI
MOV CX, 100          ; Установить счетчик элементов
REP  MOVS DST, SRC    ; Скопировать байты.
```

Обычно ассемблер (в зависимости от версии) допускает использование для передачи цепочки мнемокоды MOVSB (move byte string) или MOVSW (move word string), которые определяют тип элементов строки 8-байт w-слово. При этом операнды в команде могут отсутствовать.

Команды сравнения строк CMPS (*compare string*). Формат команды имеет вид:

```
CMPS SRC, SRC   ; < SRC > = < DST >  ?
```

Эта команда производит вычитание байта или слова строки DST, адресуемой регистром DI из байта или слова цепочки SRC, адресуемой регистром SI. В зависимости от результата вычитания устанавливаются флажки, но сами операнды не изменяются.

Когда перед CMPS находится префикс REPE или REPZ операция интерпретируется как «сравнивать, пока не достигнут конец строки», (или пока элементы строки не будут равны).

При наличии префикса REPNE или REPNZ операция интерпретируется как «сравнивать, пока не достигнут конец строки» (или пока элементы строки будут равны).

Например, команды

```
CLD
MOV CX, 100
REPE    CMPS DST, SRC
```

будут сравнивать до 100 пар элементов строк SRC - DST, пытаясь найти два несовпадающих элемента.

Команды

```
CLD
MOV CX, 100
```

REPNE CMPS DST, SRC

будут сравнивать до 100 пар элементов строк SRC и DST, пытаясь найти два совпадающих элемента. Для явного указания типа элементов строки обычно допускаются мнемокоды CMPS и CMPSW, при этом операнды в команде отсутствуют.

Команда сканирования SCAS имеет формат:

SCAS DST ; <ac> - <dst>

По этой команде вычитается элемент строки DST (байт или слово), адресуемое регистром DI, из содержимого аккумулятора AL или AX. В соответствии с полученной разностью устанавливаются флажки, но значение операндов не изменяется.

Для выполнения действий более чем над одним элементом строки, надо воспользоваться префиксами повторения REPE (REPZ) или REPNE (REPNZ).

Например, с помощью последовательности команд

```
        CLD
        LEA   DI,  ES: B_STRING
        MOV AL,
        MOV CX, 100
REPE    SCAS B_STRING
```

можно просмотреть до 100 элементов строки байтов B_STRING в поисках элемента, отличного от пробела. Для явного указания типа элементов строки, обычно используются мнемокоды SCASB и SCASW.

5.4.4. Команды безусловной и условной передачи управления

Команды передачи управления используются в разветвляющихся и циклических программах, а также при вызове подпрограмм и возврате из них.

Сегментная организация памяти определяет два основных типа команд передачи управления: внутрисегментные NEAR (близкие) и межсегментные FAR (дальние). При выполнении команды типа NEAR модифицируется только регистр IP, и адрес переходов представляется одним словом и даже байтом, если используется короткий вариант перехода с ограниченным диапазоном адресов. При выполнении команды типа FAR изменяется содержимое регистров IP и CS и адрес перехода представляется двумя словами (сегмент:смещение), что позволяет перейти в любую точку адресного пространства памяти.

Ассемблер поддерживает большое число команд условного перехода, которые осуществляют переход в зависимости от состояния флагового регистра. Существуют знаковые и беззнаковые команды условного перехода.

Использование команд определяется типом данных, над которыми производятся операции. В рассматриваемую группу команд входят команды безусловных и условных переходов, вызовов, возвратов, управления циклами и прерываний.

Команды безусловных переходов JMP производят модификацию регистра IP или регистров IP и CS без сохранения прежних значений этих регистров. Имеется три формата команды JMP типа NEAR, осуществляющих переход в пределах текущего кодового сегмента, и два формата типа FAR, осуществляющих переход в любую точку адресного пространства.

В двухбайтовой команде JMP dispL во втором байте содержится смещение, которое интерпретируется как знаковое целое. Это смещение добавляется (с предварительным расширением знака до 16 бит) к содержимому IP, которое соответствует адресу команды, находящейся после данной команды. Диапазон смещения dispL составляет от –128 до +127, причем при положительном смещении осуществляется переход вперед, а при отрицательном – назад. Данная команда реализует короткий переход, который в мнемоническом обозначении отмечается указателем SHORT (например, JMP SHORT COUNT – короткий переход к метке).

Трехбайтная команда JMP disp производит такое же действие, как и предыдущая команда, но содержит 16-битовое смещение (ассемблерный указатель NEAR PTR). Это смещение интерпретируется как знаковое число в диапазоне –32 768 до +32 767, что обеспечивает переход в любую точку кодового сегмента.

Команда JMP r/m осуществляет косвенный внутрисегментный переход (ассемблерный указатель WORD PTR), при котором в регистр загружается содержимое регистра или ячейки памяти.

Команда JMP m реализует косвенный межсегментный переход (ассемблерный указатель DWORD PTR) через содержимое двух ячеек памяти.

Команда JMP addr реализует прямой межсегментный переход (ассемблерный указатель FAR PTR) и содержит 4 байта адреса перехода: два байта сегментное смещение загружается в IP, а два байта – в регистр CS.

Команды вызова подпрограмм CALL имеют такие же форматы, как и команда JMP и выполняются аналогичным образом, за исключением того, что автоматически запоминается адрес возврата (т.е. адрес команды, следующий за командой CALL). С этой целью при внутрисегментных вызовах в стеке запоминается содержимое регистра IP, а при межсегментных вызовах – сначала содержимое IP, а затем CS.

Команды возвратов RET из подпрограмм возвращают управление программе, осуществившей вызов. Такая передача управления осуществляется путем извлечения из стека адреса возврата. Поэтому команды возврата не содержат никакой адресной информации.

Тип команды возврата выбирается в соответствии с типом команды CALL, осуществивший вызов данной подпрограммы. Каждая подпрограмма должна обязательно содержать команду возврата. Команды условных переходов осуществляют передачу управления в зависимости от значения флаговых регистров. Имеется 18 команд условных переходов, которые представлены единым двухбайтовым форматом, позволяющим осуществлять короткие (в пределах от –128 до +127) переходы относительно указателя команд IP.

Команды условных переходов обеспечивают ограниченный диапазон переходов, для расширения которого необходимо сочетать их с командами безусловных переходов. Перечень команд условного перехода приведен в табл.5.4.1.

Таблица 5.4.1 - Команды условного перехода

Команда	Описание	Условие перехода
JA	Jump if Above – перейти, если выше	CF=0, ZF=0
JAE	if Above or Equal…, выше или равно	CF=0
JB	if Below…, если ниже	CF=1
JBE	if Below or Equal …, ниже или равно	CF=1 ,ZF=1
JC	if Larry…, если перенос	CF=1
JCXZ	if CX is Zero, если значение СЧ равно 0	CX=0
JE	if Equal…, если равно	ZF=1
*JG	if Greater…, если больше	$ZF=0_n$ $SF_r=0_f$
*JGE	if Greater or Equal, если больше или равно	SF=0F
*JL	if Less, если меньше	SF≠ZF
*JLE	if Less or Equal, если меньше или равно	ZF=1 SF≠0
JNA		CF=1 ZF=1

JNAE	if Not Above, если не выше	CF=1
JNB	if Not Above nor Equal, не выше и не равно	CF=0
JNBE	if Not Below, если не ниже	CF=0, ZF=0
JNC	, не ниже и не равно	CF=0
JNE	if No Carry, если нет переноса	ZF=0
*JNG	if Not Equal, если не равно	ZF=1, SF≠0
*JNGE	if Not Greater, если не больше	SF≠0
*JNL	if Not Greater nor Equal, не больше и не равно	SF=0
*JNLE	if Not Less, если не меньше	ZF=0, SF=0
*JNO	, не меньше и не равно	OF=0
JNP	if No Overflow, если нет переполнения	PF=0
*JNS	if No Parity, если нет четности	SF=0
JNZ	if No Sign, если знаковый разр.33 нулевой	ZF=0
*JO	if No Zero, если не нуль	OF=1
JP	if Overflow, если переполнение	PF=1
JPE	if Parity, если четное число бит	PF=1
JPO	if Parity Even, если сумма битов четная	PF=0
JS	if Parity Odd, если сумма бит нечетная	SF=1
JZ	on Sign, если знак бит =1	ZF=1
	if Zero перейти, если нуль	

Команды со значком *- относятся к действиям над числами со знаком (в обратном коде).

Как видно из таблицы, большинство команд можно кодировать одним из двух мнемонических кодов. Например, JB и JNAE генерируют один и тот же объектный код, хотя положительную проверку JB понять легче, чем отрицательную JNAE.

Для беззнаковых чисел есть переходы по состояниям

Равно Equal

Выше Above

Ниже Below,

а для знаковых

Равно Equal

Больше Great

Меньше Less.

Переход по паритету JP / JPE приводит к передаче управления по определенному адресу, если в результате операции обнаружен четный паритет, т.е. в младших 8 битах имеется четное число единиц.

5.4.5. Команды управления циклом и микропроцессором

Команды управления циклом используются для удобства реализации вычислительных циклов (итераций). Они осуществляют условный переход в зависимости от состояния регистра CX, который выполняет функции счетчика циклов. Команды LOOP, LOOPE и LOOPNE, кроме того декрементируют регистр CX перед выполнением условного перехода, что исключает необходимость использования команды декрементации. Как и в командах условных переходов, при выполнении условия управление передается команде, расположенной в диапазоне адресов от –128 до +127, задаваемых смещением во втором байте команды.

Команда LOOP, которая обычно ставится в конце цикла, осуществляет декремент CX и если CX≠ 0, то добавляет смещение к регистру IP, в противном случае (CX=0 и цикл окончен) выполняется следующая по порядку команда.

Команда JCXZ имеет противоположный смысл и осуществляет переход только при CX=0. Ее удобно использовать в начале цикла, особенно в ситуации, когда цикл может не выполняться ни разу. Если команда LOOP выполняет постпроверку содержимого CX, то команда JCXZ выполняет его предпроверку.

Команда LOOPE/LOOPZ по сравнению с командой LOOP вводит дополнительное условие повторения цикла: ZF=1. Это позволяет выходить из вычислительного цикла как по окончании заданного цикла, так и получении ненулевого результата или неравенстве сравниваемых операндов.

К группе команд управления процессором относятся команды ESC, HLT, LOCK, NOP и WAIT.

ESC применяется при выполнении специальных операций над числами с плавающей точкой и обеспечивает переключение центрального процессора на

сопроцессор. ЦП передает в сопроцессор инструкцию и операнд для выполнения необходимой операции.

HLT вызывает останов процессора, при котором происходит ожидание прерывания. При завершении команды HLT регистры CS:IP указывают на следующую команду. При возникновении прерывания процессор записывает в стек состояние регистров CSи IP и выполняет подпрограмму обработки прерывания. При возврате из подпрограммы команда IRET восстанавливает регистры CS и IP из стека и управление передается на команду, следующую за HLT.

LOCK осуществляет блокировку шины доступа к данным. Запрещает другим (со)процессорам одновременно изменять элементы данных. Эта команда представляет собой однобайтный префикс, который можно кодировать непосредственно перед любой командой. В результате выполнения команды LOCK в другой процессор посылается сигнал, запрещающий использование данных, пока не будет завершена команда.

NOP применяется для удаления или вставки машинных кодов или для задержки выполнения программы. Команда NOP выполняет операцию XCHG AX,AX, которая не меняет содержимое регистров и состояние процессора.

WAIT позволяет процессору оставаться в состоянии ожидания, пока не произойдет внешнее прерывание. Данная процедура необходима для обеспечения синхронизации процессора с внешним устройством или сопроцессором. Процессор ожидает, пока внешнее устройство (или сопроцессор) не закончит выполнение операции и на входной линии TEST не появится активный уровень.

5.5. Способы адресации МП К1810

Команды микропроцессора ВМ86 реализуют разнообразные способы адресации, что упрощает организацию и использование сложных структур данных, а также расширяет возможности отдельных команд и повышает гибкость их применения.

1. **Регистровая адресация**. Операнд находится в одном из общих регистров МП, а в некоторых командах в одном из сегментных регистров.

Примеры:

```
MOV AX, SI  ; <SI>→<AX>
ADD DI, BX  ;  <BX>+<DI>
AND CL, AX  ;  Ошибка, несоответствия размеров регистров
XOR AL, AH  ; XOR <AL> и <AH>.
```

2. **Непосредственная адресация**. Операнды представляют собой константы длиной 8 или 16 бит, содержащиеся в командах. В МП нет команд непосредственной загрузки регистров.

```
SUB AL, 30H          ; <AL>- 48   (30H = 48D)
```

```
MOV CL, 10              ;  (10→<CL>)
AND AX, 0F000H      ; Выделить старших 4 бита в AX
XOR DH, 1               ; Инвертировать младший бит в  DH
CMP BL,40H            ; Сравнить содержимое BL с числом 64.
```

3. **Прямая адресация.** Эффективный адрес берется из поля смещения команды:

```
MOV AX, GAMMA    ;  В Акк AX загружено содержимое ячеек
памяти
                 ;  с адресом, полученным суммированием
                 ;  <DS> сдвинутого на 4 разряда с адресом
                 ;  переменной GAMMA, определенном в
ячейке программы.
ADD TEMP, BL     ;  <BL>+ <<DS>↑4+TEMP>
```

4. **Косвенная регистровая**. Эффективный адрес (EA) находится в одном из базовых или индексных регистров. В косвенной адресации могут использоваться только регистры BX, SI, DI. Косвенные регистровые операнды заключают в квадратные скобки.

```
ADD AX, [DI]  ;  К <AX> прибавляется содержимое ячейки
памяти,
              ;  адрес который находится  в DI.
              ;  исполнительный адрес: <DS>↑4+ <DI>.
MOV [SI], CL  ; <<SI>> ←< CL>
```

5. **Базовая адресация** (база + смещение). EA определяется суммой значения смещения, указанного в команде и содержимого регистров BX или BP.

К операциям в памяти можно адресовать, указывая *прямой адрес,* т.е. называя имя соответствующей области памяти, либо *косвенные* - через регистры-указатели, или индексные регистры. При прямой адресации 16-ричное смещение автоматически складывается с базовым адресом соответствующего сегмента. При косвенном обращении участвуют один или два из четырех регистров. Они указываются в квадратных скобках [ ] – признак косвенной адресации BX, BP, SI, DI. Если указывается переменная, за ее именем следует выражение в квадратных скобках, которое задает базовые или индексные регистры.

В случае косвенной адресации может быть указан либо только базовый регистр, либо только индексный, либо оба регистра и может быть также указано 8 или 16-битовое смещение.

```
MOV AX, [BX]    ;  переслать слово из памяти в AX. Слово находится
в сегменте
                ;  данных, адрес этого сегмента в регистре DS, смещение
```

; относительно этого адреса в регистре BX.

К операндам, находящимся в памяти, можно обратиться одним из четырех способов:

Указанием прямого 16-разрядного смещения

MOV REPORT, AL ;в байт памяти с именем REPORT пересылается содержимое AL

; ([DS]$\uparrow^4$+[REPORT])$\rightarrow$AL

Использованием косвенного обращения через базовый регистр, содержимое которого суммируется с 8 или 16-разрядным смещением.

MOV ON[BX+2], AL

MOV BL, ON[BP]

Использованием косвенного обращения через индексный регистр, содержимое которого суммируется со смещением

MOV CL, ITEM[SI+1]

MOV ON[DI+1], CL

Использованием косвенного обращения через базовый и индексный регистры, содержимое которых суммируется со смещением

MOV AH, ITEM[BX+1] [SI+1]

MOV ON[BX+1][DI+1]

При определении BP в качестве базового регистр обращение осуществляется к текущему сегменту стека SS (если нет префикса замены сегмента). Это делает базовую адресацию с регистром BP очень удобным средством обращения к данных, находящимся в стеке. Обычно базовый регистр BREG указывает на начало структуры данных, а требуемый элемент адресуется с помощью смещения (расстояния) от базы.

Для обозначения базовой адресации используют два представления:

1) [BREG] DISP BREG – базовый регистр (BX или BP)
2) [BREG+ DISP]

Пример записи команд с базовой адресацией:

MOV AX, [BP]10 ; Обе команды передают шестое слово массива, адресуемое BP,

MOV AX, [BP+10] ; в Аккумулятор,

ADD [BX]TEMP,CX ; Прибавить <CX> к слову TEMP в массиве, адресуемом BX.

6. **Индексная адресация** (смещенная база + индекс). Этот вид адресации называют адресацией с индексированием. Эффективный адрес вычисляется как сумма смещения, находящегося в команде, и содержимого индексного регистра DI или SI. Адресация удобна при доступе к элементам таблицы

(массива), когда смещение указывает на начало таблицы (массива), а индекс – на элементы в таблице (массиве). По существу индексная и базовая адресация в МП К1810 аналогичны. Это объясняется тем, что базовые и индексные регистры имеют одинаковую длину. Индексная адресация обозначается в виде TABL[IREG]. Здесь TABL – 16-битовое смещение (адрес начала таблицы).

Примеры:

```
MOV ADRM [SI], AX   ; Передать <AX> в элемент массива с начальным
                    ; адресом ADRM
ADD CX, MASS [DI]   ; Прибавить к <CX> элемент массива.
                    ; MASS – смещение, указанное в команде.
```

Пример: Загрузить 3-й элемент массива в аккумулятор AL.

```
table DB 10, 20, 30, 40
MOV DI, 2           ; загрузить в индексный регистр номер выбираемого байта
                    ; минус 1 (т.к. массив начинается с нулевого элемента)
MOV AL, TABLE [DI] ; загрузить 3-й байт таблицы в AL.
```

Примеры записи базового индекса адресации:

```
MOV AX, [BX+2+DI]  ; Операнды можно заключать в скобки
MOV AX, [DI+BX+2]  ; в любом порядке, а сдвиг можно
MOV AX, [BX+2][DI]  ; сочетать с любым из регистров
MOV AX, [BX][DI+2]
```

7. **Базово-индексная адресация** (по базе с индексированием). Эффективный адрес равен сумме содержимого базового регистра, индексного регистра и, возможно, смещения, указанного в команде. Этот способ целесообразно использовать при работе с двумерными таблицами. В этом случае базовый регистр содержит начальный адрес массива, а значения смещения и индексного регистра является смещением по строке и столбцу. В ассемблере МП 1810ВМ86 базово-индексная адресация представляется в виде: [BREG]ADR16[IREG].

Пример: загрузить в AX 16-разрядный элемент таблицы, состоящей из 4-х столбцов и 3-х строк, находящийся в третьей строке на третьей позиции (3,3) [a_{22}, если в таблице считать 0-й столбец и 0-ю строку].

```
TABLE DW 1024, 1048, 2048, 3600    ; Задание таблицы в начале ассемблерной
     DW 4100, 5000,   600, 2000        ; программы
     DW     80,   300, 4000, 5000        ;
     VALUE DB 2          ; указание номера элемента в строке минус 1 (т.к. считается с 0).
     ……..
     MOV BX, TABLE ; в базовый регистр BX заносится начальный адрес таблицы
     MOV DI, 16          ; в индексный регистр DI заносится смещение в байтах (ячейка
                         ; памяти) адреса начала третьей строки от начала таблицы.
     MOV BX, VALUE [BX][ DI] ; загрузка элемента (a22)=4000 в AX.
```

8. **Относительная адресация**. Эффективный адрес вычисляется как сумма фиксированного смещения, находящегося в команде и текущего значения программного счетчика PC. При этом значение PC равно адресу байта, следующего за текущей командой. В МП К1810 относительная адресация применяется только в командах условных и безусловных переходов, вызова подпрограмм и управления итерациями (или циклами). Следует отметить, что программист в ассемблерных программах указывает не значение смещения, а абсолютный адрес перехода, т.е. метку команды, которой необходимо передать управление. Значение смещения выполняется автоматически программа - ассемблер.

9. **Адресация цепочек**. Для обращения к операциям цепочечных команд, используются индексные регистры. Регистр SI адресует первый байт (слово) цепочки источника, а регистр DI – первый байт (слово) цепочки получателя. В повторяющихся цепочных операциях МП автоматически изменяет содержимое регистров SI и DI.

10. **Адресация портов ввода-вывода.** Существует прямая и косвенная адресация портов. В прямой адресации номер порта представляет собой 8-битовый непосредственный операнд, находящийся во втором байте команды, что обеспечивает обращение к фиксированным портам 0-255.

При косвенной адресации номер порта находится в регистре DX и имеет диапазон 0-65535. С помощью предварительной инициализации регистра DX одна и та же команда может обращаться к любому порту в адресном пространстве ввода-вывода.

Примеры:

```
IN AL, 40H    ; Ввод байта из порта номер 40H
```

```
OUT DX, AX ; Вывод слова в порт с адресом, хранящемся в DX
IN AX, DX  ; Ввод слова из устройства, адрес которого хранится
```