Министерство общего и профессионального образования Российской федерации

Ульяновский государственный технический университет

ФУНКЦИОНАЛЬНАЯ ОРГАНИЗАЦИЯ МИКРО-ЭВМ И МИКРОКОНТРОЛЛЕРОВ

Часть 2: К1816ВЕ48

Методические указания для студентов направления "Информатика и вычислительная техника"

Составители: В.Н.Негода
И.А.Никищенков

Ульяновск 1996

УДК 681.3(076)

Функциональная организация микро-ЭВМ и микроконтроллеров. Часть 2: К1816ВЕ48. Методические указания для студентов направления"Информатика и вычислительная техника"/Сост. В.Н.Негода, И.А.Никищенков. - Ульяновск:, УлГТУ, 1996. - 32 с.

Настоящие методические указания написаны в соответствии с рабочими программами дисциплин "Микропроцессорные системы", "Функциональная организация ЦВМ" и "Специализированные вычислительные системы и комплексы" для студентов направления "Информатика и вычислительная техника". Представлены справочные материалы по функциональной организации микроЭВМ К1816ВЕ48.

Приведенный материал ориентирован на выполнение различного рода заданий в ходе лабораторных и практических занятий, курсового проектирования, и сдачи экзаменов. Для всех команд приводится алгоритмическое описание их выполнения в ЭВМ.

Подготовлены на кафедре "Вычислительная техника".
Ил.2, табл. 1, библиогр.: 2 назв.

Рецензент:

Одобрено учебно-методической
комиссией ФИСТ

Оглавление

Введение

Рабочие программы учебных дисциплин "Микропроцессорные системы", "Функциональная организация ЦВМ" и "Специализированные вычислительные системы и комплексы" для студентов направления "Информатика и вычислительная техника" предусматривают изучение архитектур разнообразных микроконтроллеров (МК), микропроцессоров (МП) и ЭВМ на их основе. При выполнении лабораторных заданий, контрольных работ, курсовых проектов и решении экзаменационных задач приходится использовать большой объем фактологического материала. Производительнось учебной деятельности при этом во многом зависит от доступности и понятности студентам справочных данных по форматам и системе команд, способам адресации и программированию на ассемблере. Составители настоящих методических указаний попытались сформировать справочные данные по различным архитектурам на основе единого стиля описания, что позволяет сократить время на вхождение в архитектуры различных МК, МП и ЭВМ. Во второй части рассматривается функциональная организация однокристальной микроЭВМ К1816ВЕ48 (далее называемой микроконтроллер или МК), которая является функциональным аналогом семейства MSC-48 фирмы Intel. В семейство MSC-48 входит десять микроконтроллеров, которые отличаются друг от друга различной конфигурацией ЗУ программ и данных, размещаемых на кристалле. Его развитием является семейство MSC-51, которое содержит более обширное адресное пространство памяти программ и данных, обладает рядом других преимуществ и включает в себя 41 микроконтроллер. В России в составе серий К1816 и К1830 реализовано 6 микроконтроллеров семейства MSC-48 и 5 микроконтроллеров семейства MSC-51. Система команд всех микроконтролеров семейства MSC-48 одинакова и совместима снизу вверх с системой команд семейства MSC-51, поэтому изучение микроконтроллера К1816ВЕ48 позволяет изучить особенности построения промышленных микроконтроллеров.

Опыт преподавания вопросов организации ЭВМ и МП в различных дисциплинах показывает, что подавляющему большинству студентов для осмысления архитектуры требуется выполнение достаточно большого объема различного вида практических работ, основными из которых являются:

- ручное ассемблирование, дизассемблирование и анализ машинных программ;
- программирование на языках ассемблера;

- моделирование МК, МП и микропроцессорных устройст и систем.
Содержание методических указаний ориентировано на выполнение этих видов работ.

1. Основные термины и язык описания архитектуры

1.1. Основные понятия

В различных справочниках, учебниках и инженерных изданиях используются разные определения основных понятий, применяемых для описания архитектуры МП, МК и ЭВМ. Приводимый ниже список определений понятий не претендует на какую-то большую точность, строгость или полноту. Правильнее воспринимать этот список как соглашение типа "в данной работе это трактуется так", поэтому читатель должен быть готов к тому, что в других изданиях он может встретить несколько иные толкования.
Функциональная организация МП, МК или ЭВМ - совокупность программно-доступных компонентов, способов адресации, форматов и наборов команд.
Программно-доступные компоненты - любые объекты, содержимое которых может быть использовано или модифицировано с помощью программы.
Машинная программа (machine code) - набор данных в памяти ЭВМ, определяющий выполнение программы процессором ЭВМ.
Машинная команда (instruction) - минимальная единица машинной программы, представляющая собой совокупность данных, определяющих работу процессора при выполнении одной операции.
Адрес команды - адрес первого байта машинной команды.
Счетчик команд (PC - Program Counter, instruction counter, instruction pointer) - регистр процессора, где формируется адрес команды; обычно по мере выборки из памяти частей команды значение счетчика увеличивается на количество прочитанных байтов.
Формат команды (instruction format) - совокупность полей (групп разрядов) машинной команды с указанием местоположения и смысла данных, представляемых каждым полем.
Код операции (КОП, OpCode) - поле команды, определяющее операцию, которая должна быть выполнена по данной команде.
Операнд (operand) - данное, используемое при выполнении команды. Наиболее важными параметрами операнда являются длина (обычно в байтах) и местоположение (регистр процессора, ячейка памяти, порт ввода-вывода).

Операнд-источник - операнд, исходное значение которого используется при выполнении команды.
Операнд-приемник - операнд, местоположение которого совпадает с местоположением результата операции.
Методы адресации (addressing schemes) - методы определения местоположения операнда. Метод адресации может представляться в отдельном поле команды или неявно задаваться кодом операции.
Адресное поле - поле команды, используемое для определения местоположения операнда. Местоположение одного операнда может быть задано содержимым нескольких адресных полей: полем метода адресации, полем номера регистра, хранящего операнд или адрес, полем смещения, используемого в качестве слагаемого для вычисления адреса.
Эффективный адрес (EA - Effective Adress) - адрес операнда, вырабатываемый при обработке адресного поля команды в соответствии с заданным методом адресации. Иногда эффективный адрес задается неявно без использования адресных полей.
Физический адрес - адрес ячейки памяти, где находится операнд. В ЭВМ, где есть аппаратура поддержки распределения памяти между программами, физический адрес формируется из эффективного адреса и набора базовых адресов, указывающих на местоположение данных выполняемой программы.
Регистровая адресация - в адресном поле указывается номер регистра, где находится операнд.
Абсолютная (прямая) адресация (absolute addressing, direct addresing) - в адресном поле указывается EA.
Непосредственная адресация (immediate addresing) - в адресном поле приводится значение операнда.
Смещение (displacement, offset) - содержимое адресного поля, используемое как слагаемое для формирования адреса в индексной адресации.
Индексная адресация (indexed addresing) - EA определяется как сумма смещения и содержимого регистра.
Базовая адресация (base addresing) - аналогично индексной. Базовая и индексная адресации обычно различаются по смыслу обрабатываемых группой команд данных и способам организации доступа к ним.
Косвенный адрес (IA - Indirect Adress, deferred address) задает адрес, по которому находится EA.
Косвенная адресация (indirect addresing) - адресные поля операнда в соответствии с методом адресации определяют косвенный адрес. Косвенная адресация может сочетаться со многими другими адресациями.

Косвенная регистровая адресация - эффективный адрес EA находится в регистре, номер которого задан в адресном поле команды. При этом данный регистр выполняет функции адресного. В некоторых архитектурах функции адресного регистра могут выполнять только вполне определенные регистры. Имеются архитектуры, где любой регистр может выполнять функции адресного.

Неявная адресация (implied addresing, inherent addresing) - адресация, при которой местоположение операнда задается кодом операции без использования адресных полей.

Стек (stack) - область памяти, обращение к которой выполняется через стековую адресацию.

Стековая адресация - адресация с использование специального регистра - указателя стека (SP - Stack Pointer). Стековая адресация используется для занесения операндов в стек в одном порядке и извлечения в обратном порядке. До занесения операнда Op в стек содержимое SP уменьшается, затем используется в качесте EA операнда-приемника. При извлечении Op из стека в качестве EA операнда-источника используется содержимое SP, затем SP увеличивается. Таким образом, значение SP всегда указывает на последнее данное, находящееся в стеке. Стековая адресация часто бывает неявной - например в командах входа в подпрограмму и выхода из подпрограммы, где операндом является адрес возврата. В некоторых ЭВМ имеется возможность организовать несколько стеков, используя в качестве указателей адресные регистры.

Косвенная регистровая адресация с автоувеличением или автоуменьшением содержимого регистра - адресация, при которой обращение к операнду по EA, равному содержимому используемого регистра, сопровождается автоматическим уменьшением или увеличением его содержимого на длину операнда. Различают предмодификацию (изменение содержимого регистра до обращения к операнду) и постмодификацию (изменение после обращения). Взаимнопротивоположный порядок модификации для адресации с автоувеличнием и автоуменьшением позволяет организовать стековую адресацию на основе любого регистра.

Адрес входа (entry point) - адрес первой команды подпрограммы или программы обработки прерывания.

Адрес возврата (return address) - адрес команды, перед выполнением которой вызвана подпрограмма или возникло прерывание.

Прерывание (interrupt) - приостановка выполнения текущей программы с возможностью после обработки прерывания

продолжить ее выполнение. При входе в прерывание всегда сохраняется адрес возврата.
Вектор прерываний (interrupt vector) - набор данных, определяющих адрес входа в подпрограмму обработки прерываний и, возможно, новые значения данных, характеризующих состояние процессора.
Адрес вектора прерываний - адрес первого байта или слова вектора прерываний.
Язык машинных кодов - запись машинной программы в виде последовательности восьмиричных или шестнадцатиричных цифр, где группа цифр задает значение байта или слова машинной программы.
Язык ассемблера (assembly language) - машинно-ориентированный язык записи программы в виде последовательности операторов, каждый из которых представляет либо директиву ассемблера, либо машинную команду. В языке ассемблера коды операции, операнды, методы адресации представляются в мнемоническом виде с использованием имен констант, переменных, адресов перехода. Для различных архитектур ЭВМ, МП и МК используются различные языки ассемблера.
Ассемблер - программа, выполняющая трансляцию программы на языке ассемблера (ассемлер-программы) в машинный код.
Мнемокод (mnemonic code) - мнемоническая запись кода операции машинной команды, используемая в языке ассемблера.

1.2. Язык описания алгоритмов выполнения команд

При описании алгоритмов выполнения команд в данных методических указаниях используются операции логических и арифметических выражений языка программирования Си:
! - логическое отрицание;
~ - поразрядное отрицание;
& - поразрядное И в двухместной операции; определение адреса в одноместной;
&& - логическое И;
| - поразрядное ИЛИ;
|| - логическое ИЛИ;
^ - поразрядное исключающее ИЛИ;
++ - увеличение на единицу для данных и на длину операнда для адресов;
-- - уменьшение на единицу для данных и на длину операнда для адресов;
- - вычитание в двухместной операции; изменение знака операнда в одноместной операции;

+ - сложение;
* - умножение в двухместной операции; обращение по адресу в одноместной операции;
/ - деление;
% - деление по модулю;
<< - сдвиг влево;
>> - сдвиг вправо;
==, !=, <, >, <=, >= - отношения равно, неравно, меньше, больше, меньше или равно, больше или равно;
= - присваивание;
-=, +=, *=, /=, %=, &=, ^=, |=, <<=, >>= - выполнение двухместной операции между левой и правой частями выражения с присвоением результата операнду, указанному в левой части;
, - операция запятая.
условие ? выраж1 : выраж2 - условное выражение; если условие истинно, то вычисляется выражение выраж1, иначе - выраж2.
Кроме перечисленных знаков операций используются обозначения:
число1..число2 - диапазон чисел от значения число1 до значения число2 включительно;
Op[номер] - значение разряда с заданным номером в операнде Op;
Op[ст] - значение старшего разряда операнда;
Op[мл] - значение младшего разряда операнда;
Op[номер1..номер2] - выделение диапазона разрядов; номер1 задает старший, а номер2 - младший разряд из указанного диапазона;
[*] - каждый разряд операнда(применить операцию к каждому разряду); например, A[*] = 1 - занести единицы во все разряды операнда A.
данное1(данное2 - конкатенация (объединение в единую последовательность) двух данных; данное1 образует старшие разряды формируемого таким образом более длинного данного;
*выражение - содержимое ячейки памяти, адрес которой определяется выражением;
Op - значение операнда;
EA - эффективный адрес;
IA - косвенный адрес;
AV - адрес вектора прерываний.

2. Архитектура однокристальной микроЭВМ К1816ВЕ48

Однокристальные микроконтроллеры семейств MSC-48 и MSC-51 широко используются в изделиях производственного и культурно-бытового назначения. Архитектура МК обеспечивает построение простых контроллеров без применения периферийных БИС. На

одном кристалле МК размещается не только узлы обработки данных, но и встроенная (резидентная) память программ и данных, порты ввода-вывода. Естественно, что при этом приходится жертвовать вычислительными возможностями микроконтроллеров, На рис.2.2. показана структурная схема МК. Основу структуры МК образует внутренняя двунаправленная 8-битная шина, которая связывает между собой все устройства БИС: арифметическо-логическое устройство (АЛУ) , устройство управления, память и порты ввода/вывода информации. Рассмотрим последовательно основные элементы структуры и особенности организации МК.

PC[7..0]
PC[11..8]

P1.0 . . . P1.7 P2.0 . . . P2.7 PB.0 . . . PB.7

Рис. 2.1.Структурная схема МК.

В состав АЛУ входят следующие блоки: комбинационная схема обработки байтов, регистры Т, регистр-аккумулятор А, схема десятичного корректора и схема формирования признаков. Аккумулятор используется в качестве регистра операнда и регистра результата. Регистр временного хранения операнда Т1 программно недоступен и используется для временного хранения второго операнда при выполнении двухоперандных команд. Комбинационная схема АЛУ может выполнять следующие операции: сложение байтов с переносом и без него; логические операции И,ИЛИ и исключающее ИЛИ; инкремент, декремент, инверсию; циклический сдвиг лево, вправо через (или минуя) флаг переноса, обмен тетрад в байте; десятичную коррекцию содержимого аккумулятора. При выполнении операций обработки данных в АЛУ на комбинационной схеме вырабатываются флаги переноса С, вспомогательного переноса (перенос из младшей тетрады в старшую) АС, нулевого содержимого аккумулятора Z и флаги наличия единицы в выбранном бите аккумулятора. Флаги переноса С и вспомогательного переноса АС фиксируются на триггерах, входящих в состав регистра слова состояния программы (ССП). Логика условных переходов по флагам, которые не сохраняются в ССП, позволяет выполнять команды передачи управления (JZ,JNZ,JB0-JB7) без их фиксации.

2.1. Программно-доступные компоненты

Регистр слова состояния программы доступен программисту и его можно прочитать (MOV A,PSW) или изменить (MOV PSW,A). Формат ССП показан на рис.2.2.

Слово состояния программы
(PSW - Program Status Word)
7
6
5
4
3
2
1
0
C
AC

F0
BS
1
S2 S1 S0

C - признак переноса;
AC - признак переноса;
F0 - флаг пользователя;
BS - селектор банка регистров;
S2S1S0 - указатель стека.

Рис.2.2. Формат ССП.

Кроме перечисленных признаков логика условных переходов МК оперирует флагами F0 и F1, функциональное назначение которых определяется разработчиком; флагом переполнения таймера TF, сигналами на входах T0 и T1. Кроме того, логикой переходов после окончания каждого машинного цикла опрашивается еще один флаг, а именно флаг разрешения/запрета прерываний.

Память программ и память данных в МК физически и логически разделены. Карта адресов памяти программ показана на рисунке 2.2. Резидентная память программ реализована в перепрограммируемом ПЗУ емкостью 1 Кбайт. Счетчик команд (СК) содержит 12 бит, поэтому максимальное адресное пространство, отводимое для программ, составляет 4 Кбайта, но для его использования необходимо организовать внешнюю память. Счетчик команд устроен таким образом, что в процессе счета участвуют только младшие 11 бит, поэтому из предельного состояния 7FFh (если только по этому адресу не расположена команда передачи управления) СК перейдет в состояние 000h. Состояние старшего бита СК может быть изменено только специальными командами (SEL MB0, SEL MB1), что обуславливает разбиение памяти на два банка памяти емкостью по 2 Кбайта каждый.

При использовании внешней памяти программ следует помнить, что МК не имеет средств считывания и анализа флага MB, равного содержимому старшего бита счетчика команд СК11. Поэтому в каждый текущий момент исполнения программы, состоящей из потока вызовов подпрограмм, нет возможности определения номера банка памяти , из которого осуществляется выборка. Так как переходы между банками выполняются только по командам SEL MB, необходимо следить за тем, чтобы подпрограммы, взаимно вызывающие друг друга, располагались в пределах одного банка памяти.

4095

. . . . . .
2048

2047

. . . . . .
1024

1023

. . . . . .
8

7
Прерывание таймера
6

5

4

3

2
Внешнее прерывание
1

0
Сброс и запуск

SEL MB1 (СК11=1)

SEL MB1 (СК11=0)

Рис. 2.2. Карта адресного пространства памяти программ.

При обработке запросов прерываний в МК старший бит счетчика команд СК11 принудительно устанавливается в 0, что приводит к необходимости размещать в пределах банка 0 подпрограмму обслуживания прерывания и все подпрограммы, вызываемые ею.
В резидентной памяти программ имеется три специализированных адреса:
адрес 0, к которому передается управление сразу после окончания сигнала СБР; по этому адресу должна находиться команда безусловного перехода к началу программы;
адрес 3, по которому расположен вектор прерывания от внешнего источника;
адрес 7, по которому расположены вектор прерывания от таймера или начальная команда подпрограммы обслуживания прерывания по признаку переполнения таймера/счетчика.
Память программ разделяется не только на банки емкостью 2 Кбайта, но и на страницы по 256 байт в каждой. В командах условного перехода задается восьмибитный адрес передачи управления в пределах текущей страницы. Команда вызова подпрограмм модифицирует 11 бит счетчика команд, обеспечивая тем самым межстраничные переходы в пределах выбранного банка памяти программ.
Резидентная память данных(РПД) емкостью 64 байта имеет в своем составе два банка рабочих регистров 0-7 и 24-31 по восемь регистров в каждом. Выбор одного из банков регистров выполняется по команде SEL RB. Рабочие регистры доступны по командам с прямой адресацией, а все ячейки РПД доступны по командам с косвенной адресацией. В качестве регистров косвенного адреса используются регистры R0, R1 и R0*, R1*.
Ячейки РПД с адресами 8-23 адресуются указателем стека из ССП и могут быть использованы в качестве 8-уровневого стека. В случае,

если уровень вложенности подпрограмм меньше восьми, незадействованные в стеке ячейки могут использоваться как ячейки РПД. При переполнении стека регистр-указатель стека, построенный на основе 3-битного счетчика, переходит из состояния 7 в состояние 0. МК не имеет команд загрузки байта в стек или его извлечения из стека, и в нем фиксируются только содержимое счетчика команд и старшая тетрада ССП (флаги). В силу этого разработчику необходимо следить за тем, чтобы вложенные подпрограммы не использовали одни и те же рабочие регистры. Карта адресов памяти программ показана на рисунке 2.3.

Практически все команды с обращением к РПД оперируют с одним байтом. Однако по командам вызова и возврата осуществляется доступ к двухбайтным словам. В памяти данных слова хранятся так, что старший байт слова располагается в ячейке с большим адресом, а в памяти программ порядок расположения байтов по старшинству при хранении двухбайтных слов обратный. В МК-системах, где используется внешнее ОЗУ, через регистры косвенного адреса R0 и R1 возможен доступ к ВПД емкостью 256 байт.

Программисту также доступен внутренний восьмибитный двоичный суммирующий счетчик, который может быть использован для формирования временных задержек и для подсчета внешних событий. Содержимое таймера/счетчика (T/CNT) можно прочитать (MOV A,T) или изменить (MOV T,A). Команда STRT T запускает таймер/счетчик в режиме таймера, а команда STRT CNT- в режиме счетчика событий. Остановить работу (но не сбросить содержимое) таймера/счетчика можно командой STOP TCNT или сигналом системного сброса СБР.

63

32

Резидентная
память данных
32 * 8

31

26

Банк
регистров 1
25
R1*
24
R0*
23

8

8 - уровневый
стек или РПД
7

2

Банк
регистров 0
1
R1
0
R0

Рис. 2.3. Карта адресного пространства памяти данных.

Из максимального состояния FFh таймер/счетчик переходит в начальное состояние 00h. При этом устанавливается в 1 флаг переполнения, что вызвает прерывание, если оно разрешено командой EN TCNTI. Если прерывания запрещены командой DIS TCNTI флаг переполнения может быть опрошен по команде условного перехода JTF. Выполнение команды JTF, как и переход к подпрограмме обработки прерывания по вектору с адресом 7, сбрасывает флаг переполнения таймера/счетчика. В режиме таймера на вход таймера/счетчика через делитель частоты на 32 поступают синхросигналы машинного цикла (частота которых определяется как fcy/15, где fcy - частота генератора тактовых сигналов МК) и при fcy= 6 Мгц счетчик увеличивает свое состояние на 1 через каждые 80 мкс (12,5 КГц). В режиме счетчика событий внутренний счетчик увеличивает свое состояние на 1 каждый раз, когда сигнал на входе T1 переходит из высокого к уровня к низкому. Минимально возможное время между двумя входными сигналами равно 3 машинных цикла (что равно7,5 мкс при использовании резонатора 6 МГц). Минимальная длительность высокого уровня на входе T1 составляет 1/5 машинного цикла, а низкого уровня - машинный цикл.

2.2. Форматы команд

Все команды имеют формат один или два байта (70% команд однобайтные). Первое слово определяет операцию, длину операнда и методы адресации. Второе слово всегда является адресным и может содержать непосредственный операнд или абсолютный адрес. Основными являются следующие форматы команды:

Первый байт
Второй байт

7
6
5
4

3
2
1
0
7
6
5
4
3
2
1
0
Тип 1
Op Code
Тип 2
Op Code
#d
Тип 3
a10 a9 a8
Op Code
a7
a6
a5
a4
a3
a2
a1
a0
Тип 4
Op Code
ad8

Здесь:Op Code - код операции;
#d - непосредственный операнд;
ad8 - адрес в пределах текущей страницы;
a10..a0 - адрес в пределах текущего банка.

2.3. Способы адресации

Для обращения к данным используются четыре способа адресации:

1. Регистровая(разработчики МК называют ее прямой адресацией),
когда адрес операнда содержится в теле самой команды. Операндом в этом
случае является регистр, причем его номер указывается в трех младших
битах кода операции (Op Code). В мнемонике ассемблера используются
обозначения R0..R7, R0'..R7'.
2. Непосредственная, когда сам 8-битный операнд (константа) располагается во втором байте команды. В ассемблер-программе перед
непосредственным операндом ставиться знак "#".
3. Косвенно-регистровая(разработчики МК называют ее просто
косвенной), при которой адрес операнда располагается в регистре (R0
или R1), причем его номер указывается в младшем бите кода операции (Op
Code).
4. Неявная, при которой в коде операции содержится неявное указание
на один из операндов.Чаще всего таким операндом является аккумулятор.

2.4. Система команд

Система команд МК включает в себя 96 основных команд и ориентирована на реализацию процедур управления. Время выполнения команд составляет один или два машинных цикла (2,5 или 5,0 мкс при тактовой частоте 6,0 Мгц). Большинство команд выполняется за один машинный цикл.За два машинных цикла выполняются команды с непосредственным операндом, ввода/вывода и передачи управления. Набор команд МК невелик. Для большинства команд допустимы только определенные способы адресации операндов и используемых при этом регистров, причем при этом меняется код команды, что затрудняет изучение системы команд МК и увеличивает таблицу кодов команд.
Система команд МК и правила их выполнения допускают обработку следующих типов данных:
- символы (байты);
- целые числа без знака из диапазона 0..127 (тетрады);
- целые числа без знака из диапазона 0..255 (байты);

- обработка отдельных разрядов (битовых полей).

В описании команд, приводимом ниже, используется двоичное представление кодов команд. Символ n в коде команды обозначает использование в качестве операнда содержимого регистра, номер которого от 0 до 7, символ r - использование косвенной адресации через регистры 0 или 1. Символ p=1..2 используется для задания номера порта ввода/вывода, а символ s=4..7 определяет номер дополнительного порта ввода/вывода.

Все команды пересылки данных (таблица 1), за исключением MOV PSW,A не оказывают воздействия на флаги. Алгоритмы выполнения большинства команд пересылки очевидны и только некоторые команды требуют дополнительных пояснений. Так по команде MOVP A,@A содержимое ячейки текущей страницы памяти программ, адрес которой берется из аккумулятора, пересылается в аккумулятор. Аналогично по команде MOVP3 A,@A содержимое ячейки третьей страницы памяти программ, адрес которой берется из аккумулятора, пересылается в аккумулятор. При выполнении команд ввода/вывода в порты P1 и P2 IN и OUT, следует помнить, что они представляютсобой управляемые буферные регистры. При выдаче информации выводимый байт данных фиксируется в буферном регистре порта, а при вводе он не фиксируется и должен быть прочитан МК в течение периода присутствия байта на входах порта. Схемотехника портов P1 и P2 такова, что ввод данных в некоторую линию возможен только в том случае, если предварительно в данный бит порта программой МК была записана 1. Порт BUS может выполнять все функции, перечисленные для портов P1 и P2, но в отличие от них он не может специфицироать отдельные линии на ввод или на вывод. Все восемь линий порта BUS должны одновременно быть либо входными, либо выходными. По командам MOVX порт BUS используется в качестве двунаправленного синхронного канала для доступа к ВПД. Команды MOVD A,Ps и MOVD Ps,A предназначены для обмена с дополнительными портами P4..P7.

Мнемокод команды
Операнд
Машинный код
команды
Число циклов
Действие
MOV
A, Rn

11111nnn
1
A = Rn
MOV
Rn, A
10101nnn
1
Rn = A
MOV
A, @Rr
1111000r
1
A= D(Rr)
MOV
@Rr, A
1010000r
1
D(Rr) = A
MOVX
A, @Rr
1000000r
2
A = XD(Rr)
MOVX
@Rr, A
1001000r
2
XD(Rr) = A
MOV
Rn, #d
10111nnn
2
Rn = данные
MOV
@Rr, #d
1011000r
2
D(Rr) = данные
MOV
A, #d
00100011
2
A = данные

MOV
A, PSW
11000111
1
A = PSW
MOV
PSW, A
11010111
1
PSW = A
MOV
 A, T
01000010
1
A = T
MOV
 T, A
01100010
1
T = A
MOVP
 A, @A
10100011
2
A = D(PC[11..8](A)
MOVP3
 A, @A
11100011
2
A = D(0011(A)
XCH
A, Rn
00101nnn
1
A <=> Rn
XCH
A, @Rr
0010000r
1
A <=> Rr
XCHD
A, @Rr
0011000r

1
A[3..0]<=>D(Rr)[3..0]
IN
A, Pp
000010pp
2
A = Pp
INS
A, BUS
00001000
2
A = BUS
OUTL
Pp, A
001110pp
2
Pp = A
OUTL
BUS, A
00000010
2
BUS = A
MOVD
A, Ps

000011ss
2
A = 0000(Ps
MOVD
Ps, A
001111ss
2
Ps = A[3..0]

Для обработки отдельных разрядов и битовых полей используется 28 команд логических операций (таблица 2). В системе команд МК реализованы логические операции И,ИЛИ и Исключающее ИЛИ. Существует два варианта этих команд в зависимости от операндов. В первом варианте один из операндов - это аккумулятор, где и хранится результат выполнения операции. В этом случае адрес аккумулятора задается в коде команды неявно, а в качестве второго операнда может использоваться содержимое любого регистра; ячейка РПД, адресуемая через R0 или R1; константа, указываемая во

втором байте комады. Особенностью системы команд МК является второй вариант команд логических операций, где в качестве операнда-приемника используются порты ввода/вывода. Это позволяет маскировать отдельные биты портов ввода/вывода, управляя таким образом вводом данных.
Например:

```
ORL  P1,    #3,     ; разрешение ввода через биты P1[1..0]
ANL  R1,   #FB     ; очистка разряда R1[2]
 XRL R2,    #7      ; инвертирование разрядов R2[2..0]
```

Мнемокод команды
Операнд
Машинный код
команды
Число циклов
Действие
ANL
A, Rn
01011nnn
1
A &= Rn
ORL
A, Rn
01001nnn
1
A |= Rn
XRL
A, Rn
11011nnn
1
A ^= Rn
ANL
A, @Rr
0101000r
1
A & = D(Rr)
ORL
A, @Rr
0100000r
1
A |= D(Rr)

XRL
A, @Rr
1101000r
1
A ^= D(Rr)
ANL
A, #d
01010011
2
A &= данные
ORL
A, #d
01000011
2
A |= данные
XRL
A, #d
11010011
2
A ^= данные
ANL
Pp, #d
100110pp
2
Pp & =данные
ANL
BUS, #d
10011000
2
BUS @= данные
ORL
Pp, #d
100010pp
2
Pp |= данные
ORL
BUS, #d
10001000
2
BUS | = данные
ANLD
Ps, A
100111ss

2
Ps @= A[3..0]
ORLD
Ps, A
100011ss
2
Ps |= A[3..0]

Арифметические команды (таблица 3) немногочисленны и предназначены для обработки восьмибитовых целых двоичных чисел. При сложении операнд-приемник всегда аккумулятор, одресуемый неявно. Для организации вычитания необходимо перевести вычитаемое как отрицательное число в дополнительный код и сложить его с уменьшаемым. Команда сложение с учетом переноса позволяет выполнять суммирование многобайтовых чисел. Команды сдвига в качестве операнда используют только аккумулятор, который адресуется неявно, и позволяют умножить и разделить на 2 в целой степени и позиционировать разряды в нужное положение. Например:

```
; Прибавление OP1.X += OP2,
; где: X - битовое поле, размещенное в OP1[7..4],
;       OP2 - операнд
; Считается, что переполнение поля X невозможно
MOV R0,   #OP2 ; R0 =  адрес OP2
MOV A,    @R0  ; A = (OP2)
RL    A      ; позиционирование копии OP2 против поля X
RL    A      ;
RL    A      ;
RL    A      ;
MOV R0,   #OP1 ; R0 =  адрес OP1
 ADD A,   @R0  ;подсуммирование копии OP1 к полю X
```

При использовании комады перестановки тетрад программа получается короче:

```
MOV R0,   #OP2 ; R0 =  адрес OP2
MOV A,    @R0  ; A = (OP2)
SWAP A                  ; позиционирование копии OP2 против поля X
MOV R0,   #OP1 ; R0 =  адрес OP1
 ADD A,   @R0  ;подсуммирование копии OP1 к полю X
```

Код
команды
Операнд
Машинный код
команды
Число циклов
Действие
INC
A
00010111
1
A += 1
INC
Rn
00011nnn
1
Rr += 1
INC
@Rr
0001000r
1
D(Rr) += 1
DEC
A
00000111
1
Rr -= 1
DEC
Rn
11001nnn
1
Rr -= 1
ADD
A, Rn
01101nnn
1
Rr += Rr
ADDC
A, Rn
01111nnn
1
Rr += Rr+(C)
ADD

A, @Rr
0110000r
1
Rr += D(Rr)
ADDC
A, @Rr
0111000r
1
Rr += D(Rr)+C
ADD
A, #d
00000011
2
Rr += данные
ADDC
A, #d
00010011
2
Rr += данные+(C)
DA
A
01010111
1
A[3..0]>9||AC>1) ?
C(Rr=Rr + 6;
A[3..0]>9 || C>1 ?
C (Rr = Rr + 60h
RR
A
01110111
1
A = A[0])((A[7..1])
RL
A
11100111
1
Rr =(A[6..0])((A[7])
RRC
A
01100111
1
Rr((C) = (C) (Rr
RLC

A
11110111
1
(C)(Rr= Rr((C)
SWAP
A
01000111
1
Rr=A[7..0]# A[15..8]

Изменение естественного порядка следования команд выполняется по специальным командам перехода. К этой группе команд в МК относятся команды безусловного перехода JMP и JMPP, организации цикла DJNZ, обращения к подпрограмме CALL, возврата из подпрограммы RET, возврата с восстановлением PSW RETR, а также большая группа команд ветвления. Все эти команды приведены в таблицах 4 и 5. В большинстве команд прямо указывается адрес перехода, причем в теле команды при этом содержится 8 (ad8) или 11 (ad11) бит адреса перехода.
Команда JMP позволяет передать управление в любое место 2048-байтного банка памяти программ. Номер банка памяти программ определяется флагом DBF, значение которого копируется в старший бит счетчика команд (PC11) при выполнении команды JMP или CALL. Для перехода из нулевого банка памяти программ в первый недостаточно только установить флаг DBF, необходимо также выполнить команду перехода JMP, которая и изменит значение старшего бита счетчика команд.
Все остальные команды (кроме команд возврата) содержат только восемь младших бит адреса перехода. При этом оказывается возможным осуществить переход только в пределах одной страницы памяти программ (256 байт). Если команда короткого перехода расположена на границе двух страниц (т.е. первый байт команды на одной странице, а второй - на следующей), то переход будет выполнен в пределах той страницы, где располагается второй байт команды. Для услов- ного перехода с одной страницы на другую можно воспользоваться тандемом из команды условного перехода и длинного безусловного перехода (JMP).

Мнемокод команды
Операнд
Машинный код

команды
Число циклов
Действие
JMP
ad11
a10a9a810100
2
(PC[10..0]) = ad11
(PC[11]) = (DBF)
JMPP
@A
10110011
2
(PC[7..0]) = ((A))
DJNZ
Rn, ad8
11101nnn
2
(Rn) -= (Rn);
(Rn != 0) ? (PC)=ad8
CALL
ad11
a10a9a810100
2
((SP)) =
(PC):(PSW[7..4]);
(SP) ++;
(PC) = ad11;
(PC[11]) = (DBF)
RET

10000011
2
(SP) -= (SP);
(PC) = ((SP))
RETR

10010011
2
(SP) -= (SP);
(PC) = ((SP);
(PSW[7..4]) = ((SP))

Команда JMPP осуществляет переход по адресу, содержащемуся в ячейке ПП, на которую указывает содержимое аккумулятора. Таким образом, аккумулятор содержит адрес адреса перехода. Ячейка с адресом перехода должна находиться на той же странице памяти программ, что и команда перехода JMPP. Команда косвенного перехода обеспечивает простой доступ к таблице, содержащей векторы переходов по программе в зависимости от содержимого аккумулятора, что позволяет легко реализовать механизм множественных ветвлений.

Для организации циклов используется команду DJNZ. Счетчик циклов организуется в одном из регистров текущего банка, для чего в этот регистр загружается число повторений цикла. При выполнении команды DJNZ производится декремент и последующая проверка на нуль содержимого регистра - счетчика циклов. Если его содержимое оказывается ненулевым, то происходит переход к началу цикла, иначе - выход из цикла. Структура программы при этом будет следующей:

```
MOV     Rn, #N    ;инициализация счетчика циклов
LOOP:   ...       ;тело цикла
...
...
CYCLE: DJNZ  Rn, LOOP    ;декремент Rn и переход, если не нуль
```

Следует отметить, что команды от метки LOOP до метки CYCLE включительно должны находиться в пределах одной страницы памяти программ.

Для вызова подпрограмм используется команда CALL, позволяющая обратиться в любое место текущего банка памяти программ. При вызове подпрограммы в стеке сохраняется адрес возврата и часть ССП. Глубина вложений подпрограмм ограничена емкостью стека (16 байт) и не должна превышать восьми. Для возврата из подпрограммы необходимо выполнить команду RET, которая восстановит в счетчике команд адрес возврата. Команда RETR служит для выхода из подпрограммы обработки прерывания, так как кроме адреса возврата она восстанавливает PSW и разрешает прерывания от данного источника. При программировании прерываний МК необходимо внимательно отслеживать ситуацию вызова подпрограммы, находящейся в альтернативном (по отношению к текущему) банке ПП. В этом случае перед вызовом подпрограммы необходимо выбрать соответствующий банк памяти, а перед возвратом восстановить старое содержимое DBF.

Если в подпрограмме нет команды восстановления DBF, то возарат все же будет выполнен (т.к. в стеке сохранен полный действительный адрес возврата), однако первая же команда длинного перехода передаст управление в альтернативный банк ПП. Если к подпрограмме производятся обращения из разных банков памяти, то оказывается затруднительно восстановить значение DBF перед возвратом. В этом случае можно рекомендовать использовать команду восстановления DBF в основной программе вслед за командой вызова.

Команды ветвления обеспечивают переход от текущей команды к заданной точке программы при истинности условия (таблица 5), причем все команды условного перехода используют прямую короткую адресацию.

Из таблицы видно, что по командам условных переходов могут проверяться не только внутренние флаги, но и некоторые сигналы на внешних входах МК. Это позволяет эффективно выполнять ветвления в программе без использования процедуры предварительного ввода и последующего сравнения. Анализируемые признаки за исключением C и F0 не фиксируются в специальных триггерах флагов, а представляются мгновенными значениями сигналов в АЛУ или на соответствующих входах МК.

Если условие ветвления выполняется,то (PC[7..0]) =ad8,
иначе (PC) = (PC) +2

Мнемокод команды	Машинный код команды	Число циклов	Условие ветвления
JC	11110110	2	

C ==1
JNC
11100110
2
C == 0
JZ
11000110
2
(A) = 0
JNZ
10010110
2
(A) != 0
JF0
10110110
2
F0 == 1
JF1
01110110
2
F1 == 1
JT0
00110110
2
T0 == 1
JNT0
00100110
2
T0 == 0
JT1
01010110
2
T1 == 1
JNT1
01000110
2
T1 == 0
JTF
00010110
2
FT == 1
JNI
10000110

2
I == 0
JBi i=0..7
iii10010
2
A[i] == 1

Мнемо-код команды
Опе-ранд
Машин. код
команы
Действие
EN
I
00000101
Разрешение внешних прерываний
DIS
I
00010101
Запрещение внешних прерываний
EN
TCNTI
00100101
Разрешение прерывания по переполнению таймера/счетчика событий
DIS
TCNTI
00110101
Запрещение прерываний при переполнении таймера/счетчика событий
STRT
T
01010101
Запуск таймера
STRT
CNT
01000101
Запуск счетчика событий
STOP
TCNT
01100101

Остановка таймера/счетчика событий
ENTO
CLK
01110101
Разрешение вывода синхроимпульсов
SEL
MB0
11100101
Выбор нулевого банка (MB0) памяти программ. A[11] = 0, то есть (DBF) = 0
SEL
MB1
11110101
Выбор первого банка памяти (MB1) программ. A[11] = 1, то есть (DBF) = 1
SEL
RB0
11000101
Выбор нулевого банка рабочих регистров резидентной памяти данных
SEL
RB1
11010101
Выбор первого банка рабочих регистров резидентной памяти данных, то есть PSW[4]=1
NOP

00000000
Нет операции

Использование команд управления режимом работы (таблица 6) было рассмотрено при описании соответствующих программно доступных компонентов, за исключением команды NOP. Пустая команда NOP используется для исключения из работы фрагмента программы без модификации других фрагментов ("забой" машинных команд пустой командой) и для формирования задержки времени перед выполнением действий с периферийными устройствами.
Признаки результата могут быть изменены специальными командами установки и сброса, которые представлены в таблице 7.

Мнемокод команды

Операнд
Машинный код
команды
Число циклов
Действие
Команды установки признаков результата
CLR
A
00100111
1
(A) = 0
CPL
A
00110111
1
(A) = 0 - (A)
CLR
C
10010111
1
(C) = 0
CPL
C
10100111
1
(C) =~ (C)
CLR
F0
10000101
1
(F0) = 0
CPL
F0
10010101
1
(F0)=~(F0)
CLR
F1
10100101
1
(F1)= 0
CPL
F1

10110101
1
(F1)=~(F1)

Некоторые учебные задания по изучению функциональной организации ЭВМ предполагают анализ машинной программы с предварительным ручным дизассемблированием, т.е. формированием ассемблер-программы, соответствующей анализируемому машинному коду. При выполнении такой работы целесообразно пользоваться списком команд, упорядоченным по коду операции(таблица 7). Пример дизассемблирования фрагмента машинной программы приведен в таблице 8.

Поскольку команды с адреса 100h по адрес 10Ah представляют собой цикл суммирования массива чисел, адресуемый указателем R1, загрузка значения 10h в данный регистр перед входом в цикл позволяет сделать вывод, что это базовый адрес массива. Так как в ассемблере МК нет команд типа сравнение и вычитание, проверка адреса конца массива организуется путем сложения дополнительного кода текущего и конечного адреса массива и использования команды условного перехода JNZ по адресу 104h. Такам образом обрабатываемые данные размещены с адреса 10h по адрес 15h резидентной памяти данных.

Таблица 8. Пример дизассемблирования фрагмента программы

Адрес
Код
Мнемокод
0100
27
CLR A
0101

```
 B9
  MOV   R1,  #10H
0102
 10

0103
 61
  ADD   A,   @R1
0104
 19
  INC   R1
0105
 F9
  MOV   A,  R1
0106
 37
  CPL   A
0107
 17
  INC   A
0108
 03
  ADD   A,   #15H
0109
 15

010A
 96
  JNZ   LOOP
010B
 03

010C
 04
  JMP   M
010D
 13

010E
 B9
  MOV   R1,  #10H
010F
 10
```

```
0110
 F1
  MOV   A,  @R1
0111
 B9
  MOV   R1, #11H

 11

0113
 61
  ADD   A,  @R1
```

2.5. Ввод-вывод и прерывания

Для связи МК48 с объектом управления, для ввода и вывода информации используется 27 линий. Эти линии сгруппированы в три порта по восемь линий в каждом и могут быть использованы для вывода, ввода или для ввода/вывода через двунаправленные линии. Кроме портов ввода/вывода имеются три линии, сигналы на которых могут изменять ход программы по командам условного перехода : линия ЗПР используется для ввода в МК сигнала запроса прерывания от внешнего источника; линия Т0 используется для ввода тестирующего сигнала от двоичного датчика объекта управления; кроме того, под управлением программы (по команде ENT0 CLK) по этой линии из МК может выдаваться сигнал синхронизации; линия Т1 используется для ввода тестирующего сигнала или в качестве входа счетчика событий (по команде STRT CNT).

Специальная схемотехника портов Р1 и Р2 , которая получила название квазидвунаправленной, позволяет выполнять ввод, вывод и ввод/вывод. Каждая линия портов Р1 и Р2 может быть программным путем настроена на ввод, вывод или на работу с двунаправленной линией передачи. Для того, чтобы настроить некоторую линию на режим ввода в МК, необходимо перед этим в буферный триггер этой линии записать 1. Сигнал СБР автоматически записывает во все линии портов Р1 и Р2 сигнал 1. Квазидвунаправленная структура портов Р1 и Р2 для программиста МК 1816 специфична тем, что в процессе ввода информации выполняется операция логического И над вводимыми данными и текущими (последними) введенными данными.

Квазидвунаправленные схемы портов P1 и P2 и команды логических операций ANL и ORL предоставляют разработчику эффективное средство маскирования для обработки однобитных входных и выходных переменных.
В системе команд МК есть команды, которые позволяют выполнять запись нулей и единиц в любой разряд или группу разрядов порта , но так как в этих командах маска задается непосредственным операндом, то необходимо знать распределение сбрасываемых и устанавливаемых линий на этапе разработки прикладной программы. В этом случае, если маска вычисляется программой и заранее неизвестна, в ОЗУ необходимо иметь копию состояния порта вывода. Эта копия по командам логических операций объединяется с вычисляемой маской в аккумуляторе и затем загружается в порт. Необходимость этой процедуры вызвана тем, что в МК отсутствует возможность выполнить операцию чтения значения портов P1 и P2 для определения прежнего состояния порта вывода. Порт P2 отличается от порта P1 тем, что его младшие четыре бита могут быть использованы для расширения МК-системы по вводу/выводу. Через младшую тетраду порта P2 по специальным командам обращения возможен доступ к четырем внешним четырехбитным портам ввода/вывода P4-P7. Работа этих внешних портов синхронизируется сигналом ПРОГ.
Порт ввода/вывода BUS представляет собой двунаправленный буфер с тремя состояниями и предназначен для побайтного ввода, вывода или ввода/вывода информации. Если порт BUS используется для двунаправленных передач, то обмен информацией через него выполняется по командам MOVX. При выводе байта генерируется стробирующий сигнал ЗП, а выводимый байт фиксируется в буферном регистре. При вводе байта генерируется стробирующий сигнал ЧТ, но вводимый байт в буферном регистре не фиксируется. В отсутствие передач порт BUS по своим выходам находится в высокоимпедансном состоянии. Если порт BUS используется как однонаправленный, то вывод через него выполняется по команде OUTL, а ввод - по команде INS. Вводимые и выводимые через порт BUS байты можно маскировать с помощью команд ORL и ANL, что позволяет выделять и обрабатывать в байте отдельный бит или группу бит. В МК-системах простой конфигурации, когда порт BUS не используется в качестве порта-расширителя системы, обмен выполняется по командам INS, OUTL и MOVX. Возможно попеременное использование команд OUTL и MOVX. Однако при этом необходимо помнить, что выводимый по команде OUTL байт фиксируется в буферном регистре порта BUS, а команда MOVX уничтожает содержимое буферного регистра порта BUS. (Команда

INS не уничтожает содержимое буферного регистра порта.) В МК-системах, имеющих внешнюю память программ, порт BUS используется для выдачи адреса внешней памяти и для приема команды из внешней памяти программ. Следовательно, в таких системах использование команд OUTL лишено смысла.

В МК объекты внешних устройств (ВУ) подключаются к портаи ввода/вывода. Например, информация о достижении режущим инструментом границы допустимой зоны перемещения, получаемая при замыкании контактных пластин (так называемый "концевой выключатель"), представляет из себя один бит, который подключается к соответствующей линии порта Так же этот бит может быть включен в группу данных о состоянии устройства, например: кнопка "Стоп" инженерного пульта; двухразрядный регистр, значение в котором содержит код направления перемещения инструмента в плоскости и т.д., представляя собой слово состояния устройства, которое может занимать весь порт ввода/вывода.

Рассмотрим концевой выключатель подключенный к линии бита 3 порта 1. В этом случае процедура ожидания замыкания контакта будет иметь вид

```
WAITC:  IN    A,    P1     ;ввод сигнала от датчика
        JB3   WAITC        ;если контакт датчика разомкнут, то повторять ввод,
                           ;  иначе выход из процедуры
```

Двунаправленные порты ввода/вывода позволяют модифицировать соответствующие по командам обработки данных. Например:

```
; Зажечь светодиод, подключенный к линии бита 5 порта 2
        ORL  P2,    #20
; Погасить светодиод, подключенный к линии бита 0-го порта 1
        ANL  P1,    #FE
```

Линия запроса прерывания от внешнего источника ЗПР проверяется каждый машинный цикл во время действия сигнала САВП, но передача управления ячейке 3, где расроложена команда JMP, выполняется только по завершению цикла команды. При обработке прерывания, как и при вызове подпрограммы, содержимое счетчика команд и старшей тетрады СПП сохраняется в стеке. Ко входу ЗПР микроконтроллера через монтажное ИЛИ от схем с открытым коллектором могут быть подключены несколько источников прерывания. После распознавания прерывания все последующие запросы прерывания игнорируются до тех пор, пока по команде возврата RETR вновь будет разрешена работа логики

прерываний. Режим прерываний может быть запрещен или разрешен программой по командам DIS I и EN I. Сигнал ЗПР должен быть снят внешним устройством перед окончанием подпрограммы обслуживания, т.е. до исполнения команды RETR. В том случае, если внешнее устройство не сбрасывает свой флаг запроса прерываний при обращении МК к его буферному регистру, одна из выходных линий МК используется подпрограммой обслуживания прерывания для сброса этого флага во внешнем устройстве. Так как вход ЗПР может быть проверен по команде условного перехода JNI, то при запрещенном режиме прерывания вход ЗПР может быть использован в качестве дополнительного тестирующего входа подобно входам T0 и T1.

При необходимости в МК можно создать двухуровневую систему прерываний. Для этого надо разрешить прерывания от таймера, загрузить в него число FFH и перевести в режим подсчета внешних событий, фиксируемых на входе T1. Переход сигнала на входе T1 из состояния 1 в состояние 0 приведет к прерыванию по вектору в ячейке 7. В случае одновременного запроса прерываний от внешнего источника и запроса от флага переполнения таймера приоритет остается за источником, воздействующим на вход ЗПР.

При входе в подпрограммы обслуживания прерываний старший бит счетчика команд СК[11] принудительно устанавливается в нуль. Следовательно, вся процедура обработки прерывания должна быть размещена в банке памяти 0.

2.6. Правила записи программ и директивы ассемблера

Исходный текст программы на языке ассемблера имеет определенный формат. Каждая команда или директива представляет собой строку из четырех полей:

МЕТКА: ОПЕРАЦИЯ ОПЕРАНД(Ы) ;КОММЕНТАРИЙ

Поля могут отделяться друг от друга произвольным числом пробелов.

В поле метки размещается символическое имя ячейки памяти, в которой хранится отмеченная команда или операнд. Метка представляет собой буквенно-цифровую комбинацию, начинающуюся с буквы. Используются только буквы латинского алфавита. Длина метки не должна превышать шесть символов. Метка всегда завершается двоеточием (:). Директивы ассемблера не преобразуются в двоичные коды, а потому не могут иметь меток. Исключение составляют псевдокоманды резервирования памяти и

определения данных (DS,DB,DW). У псевдокоманд, осуществляющих определение символических имен, в поле метки записывается определяемое символическое имя, после которого двоеточие не ставится. В качестве символических имен и меток не могут быть использованы мнемокоды команд, псевдокоманд и операторов ассемблера, а также мнемонические обозначения регистров и других внутренних блоков МК.

В поле операции записывается мнемоническое обозначение команды МК или директивы ассемблера, которое является сокращением (аббревиатурой) полного английского наименования выполняемого действия. Для МК используется строго определенный и ограниченный набор мнемонических кодов. Любой другой набор символов, размещенный в поле операции, воспринимается ассемблером как ошибочный.

В поле операндов определяются операнды (или операнд), учавствующие в операции. Команды ассемблера могут быть без-, одно- или двухоперандными. Операнды разделяются запятой (,). Операнд может быть задан непосредственно или в виде его адреса (прямого или косвенного). Непосредственный операнд представляется числом (MOV A,#15) или символическим именем (ADDC A,#OPER2) с обязательным указанием префикса непосредственного операнда (#). Прямой адрес операнда может быть задан мнемоническим обозначением (IN A,P1), числом (MOV A, 40), символическим именем (MOV A,MEMORY). Указанием на косвенную адресацию служит префикс @. В командах передачи управления операндом может являться число (CALL 0135H), метка (JMP LABEL), косвенный адрес (JMPP @A) или выражение (JMP ! -2, где ! -текущее содержимое счетчика команд). Используемые в качестве операндов символические имена и метки должны быть определены, а числа представлены с указанием системы счисления, для чего используется суффикс (буква, стоящая после числа): B-для двоичной, Q-для восьмеричной, D-для десятичной и H-для шестнадцатиричной. Число без суффикса по умолчанию считается десятичным.

Поле комментария может быть использовано программистом для текстового или символьного пояснения логической организации прикладной программы. Поле комментария полностью игнорируется ассемблером, а потому в нем допустимо использовать любые символы. По правилам языка ассемблера поле комментария начинается после точки с запятой (;).

Ассемблирующая программа транслирует исходную программу в объектные коды. Хотя транслирующая программа берет на себя многие из рутинных задач программиста, таких как присвоение

действительных адресов, преобразование чисел, присвоение действительных значений символьным переменным и т.п., прграммист все же должен указать ей некоторые параметры: начальный адрес прикладной программы, конец ассемблируемой программы, форматы данных и т.п. Всю эту информацию программист вставляет в исходный текст своей прикладной программы в виде директив , которые только управляют процессом трансляции и не преобразуются в коды объектной программы.

Основные директивы:

ORG nn задает ассемблеру адрес ячейки памяти nn, в которой должна быть расположена следующая за ней команда прикладной программы;

EQU ставит в соответствие любому символическому имени, используемому в программе определенный операнд. Например, запись PET EQU 13 приводит к тому, что в процессе ассемблирования всюду, где встретится символическое имя PET , оно будет заменено числом 13;

SET переопределяет символические имена операндов, переопределяемых в процессе исполнения программы;

DB обеспечивает занесение в ПП константы, представляющей собой байт;

END указание об окончании трансляции.

В результате трансляции должна быть получена карта памяти программ, где каждой ячейке памяти поставлен в соответствие хранящийся в ней код. В соответствии с форматом команд для представления их объектных кодов отводятся одна или две ячейки памяти программ. В первой ячейке всегда располагается код операции, во второй - непосредственный операнд или адрес прямоадресуемого операнда, адрес перехода внутри страницы памяти программ.

Ниже приводится простейшая программа, которая выполняет действия:

- ввод в двоичном виде двух числовых векторов длиной по 8 элементов в массивы VECT1 и VECT2 портов P1 и P2; разрядность чисел - один байт;
- ввод чисел производится в течении часа в моменты времени, заданные в массиве TIME в формате двухбайтовых слов, причем первый байт - минуты, второй байт - секунды. По истечении часа прцедура ввода повторяется;
- формирование по месту VECT1 суммы векторов VECT1[i] += VECT[i];
- вывод полученного массива в двоичном виде в порт BUS.

```
$ALLPUBLIC

;Таблица переменных
MAXSIZE     EQU         8       ;размерность вектора

;Определение регистров 1 банка
POINTER   REQ           R0      ;указатель на вектор
COUNT       REQ         R2      ;счетчик циклов

;Определение регистров 2 банка
TABTIME     REQ         R1       ;указатель на таблицу времен
MIN1 REQ  R3     ;счетчик текущих минут
SEC1        REQ         R4     ;счетчик текущих секунд
TIK         REQ         R5     ;счетчик тиков
MIN         REQ         R6     ;минуты   табличные
SEC         REQ         R7     ;секунды  табличные



;Резервирование памяти для числовых векторов
 DEFSEG  VECTSEG,START=20H,CLASS=DATA
SEG     VECTSEG
VECT1:  DS      MAXSIZE
VECT2:  DS      MAXSIZE

;Определение таблицы времен
 DEFSEG  TIMESEG,START=30H, CLASS=DATA
 SEG     TIMESEG
 TIME:   DB      01,00,03,00,05,00,70,00

;Определение модуля основной программы
 DEFSEG  PROG,START=100H
 SEG     PROG

MAIN:       CLR         F1      ;сброс признака конца часа
        DIS         I       ;запрет внешних прерываний
        SEL         RB1
        MOV         TABTIME,#TIME;заполнение
        MOV         A,@TABTIME      ;заданных
        MOV         MIN,A           ;значений
        INC     TABTIME         ;минут и
        MOV A,@TABTIME      ;секунд
```

```
        MOV     SEC,A           ;из таблицы
        INC   TABTIME           ;TIME
        MOV     TIK,#0          ;сброс счетчика тиков
        SEL     RB0
        EN      TCNTI           ;разрешение прерываний и
        STRT    T               ;запуск таймера
WAIT:   CALL    CMPTIME         ;ожидание заданного
момента
        JF1   MAIN              ;переход по истечении часа
        JNC     WAIT
        CALL    PROCESS         ;переход к обработке данных
        JMP   WAIT              ;переход к ожиданию

;Подпрограмма обработки данных
PROCESS:  SEL       RB0
;ввод в первый вектор из порта P1
        MOV R2,#MAXSIZE
        MOV R0,#VECT1
LOOP1:  IN      A,P1
        MOV     @R0,A
        INC     R0
        DJNZ    R2,LOOP1
;ввод во второй вектор из порта P2
        MOV R2,#MAXSIZE
        MOV     R0,#VECT2
LOOP2:  IN      A,P2
        MOV     @R0,A
        INC      R0
        DJNZ     R2,LOOP2
;получение суммы и вывод в порт BUS
        MOV R2,#MAXSIZE
        MOV R0,#VECT1
        MOV     R1,#VECT2
LOOP3:  MOV     A,@R1
        ADD     A,@R0
        MOV @R0,A
        OUTL    BUS,A
        INC   R0
        DJNZ R2,LOOP3
        RET

; Обработка прерывания от таймера каждые 20,48 мсек
; и перевод текущего времени в минуты и секунды
```

```
TINT:   SEL  RB1
        INC  TIK            ;увеличение счетчика тиков и
        MOV      A,TIK      ;преобразование в
        CPL  A          ;дополнительный
        INC  A      ;код для сравнения
        ADD  A,#49          ;счетчик = 49
        JNZ  M          ;переход, если нет
        MOV      TIK,#0     ;иначе сброс счетчика тиков и
        INC  SEC1           ;увеличение счетчика секунд
        MOV  A,SEC1         ;преобразование в
        CPL  A          ;дополнительный код
        INC  A          ;счетчика секунд
        ADD  A,#60          ;тек. секунды = 60
        JNZ  M          ;переход, если нет
        MOV      SEC1,#0    ;иначе сброс счетчика секунд и
        INC  MIN1           ;увеличение счетчика минут
        MOV      A,MIN1     ;преобразование в
        CPL  A          ;дополнительный код
        INC      A          ;минут
        ADD  A,#60          ;тек. минуты = 60
        JNZ  M          ;переход если нет
        MOV  MIN1,#0        ;иначе сброс счетчика минут и
        CPL   F1            ;установить признак
                ;конца часа
M:      RETR
```

```
;Подпрограмма сравнения текущего времени с заданным
;возвращает C=1, если времена равны и C=0 - если нет
CMPTIME: JF1   C0           ;переход,если конец часа
        SEL    RB1
        MOV  A,SEC1         ;перевод счетчика
        CPL  A          ;секунд в
        INC  A          ;дополнительный
        MOV      R2,A       ;код
        MOV  A,SEC          ;и сравнение с
        ADD  A,R2           ;табличным значением
        CLR  C          ;и возврат C = 0
        JNZ  C0             ;переход если не равны
```

```
        MOV     A,MIN1          ;перевод счетчика
        CPL   A             ;минут в
        INC   A             ;дополнительный
        MOV     R2,A                ;код
        MOV A,MIN               ;и сравнение с
        ADD  A,R2               ;табличным значением
        CLR  C              ;и возврат C = 0
        JNZ   C0                ;переход если не равны
C1:         MOV     A,@TABTIME      ;подготовка новых
        MOV     MIN,A               ;значений минут и
        INC   TABTIME           ;секунд,
        MOV  A,@TABTIME     ;заданных в таблице
        MOV SEC,A               ;моментов времени для
        INC   TABTIME           ;ввода данных
        CLR  C              ;и возврат
        CPL   C             ;C = 1
C0:         RET


;Заполнение векторов прерываний
 DEFSEG  ZERO,START=0
 SEG    ZERO
        JMP   MAIN              ;Переход  к  началу
                                ;программы при запуске

DEFSEG  TINTR,START=7
SEG    TINTR
        JMP   TINT              ;Переход к программе  обработке
                            ;прерываний от таймера
        END
```

При отладке этой программы удобно использовать bat- файл:

```
avmac48.exe  prim.asm
avlink prim=prim.obj -SY
avsim48 aflprim.cmd -c1
```

в котором используется командный файл prim.cmd, подключающий к портам P1 и P2 внешние воздействия из файлов p1.dat и p2.dat :

```
laprim
IOP1.DAT
Y
```

Op1
IOP2.DAT
Y
Op2
SY

Список литературы

1. Однокристальные микроЭВМ.М.:МИКАП,1994.-400с.
2. Сташин В.А., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах.-М.:Энергоатомиздат, 1990.-224с.