ГОУ ВПО "Поволжский государственный университет телекоммуникаций и информатики"

Кафедра автоматической электросвязи

ИСПОЛЬЗОВАНИЕ МИКРОПРОЦЕССОРОВ В СИСТЕМАХ УПРАВЛЕНИЯ УЗЛОВ КОММУТАЦИЙ

Методические указания к лабораторной работе «Использование однокристальных ЭВМ серии 1816 для систем коммутации с программным управлением»

для специальности 210406, 210406у

Часть 1

Составил: к.т.н., доц. Мурашова Т.Б.

Редактор: д.т.н., проф. Карташевский В.Г.

САМАРА

2011 год

УДК 621.395

Рецензент

Заведующий кафедрой «Системы связи» ПГУТИ

д.т.н., профессор Васин Н.Н.

Одобрено
научно-методическим советом ПГУТИ

Методические указания к лабораторной работе
«Использование однокристальных ЭВМ К1816
для систем коммутации с программным управлением»
для специальности 210406, 210406у. Часть 1/под ред.
д.т.н., проф. Карташевского В.Г: Самара.– ПГУТИ, 2011 г. – 27 стр.

Лабораторная работа 1

ИСПОЛЬЗОВАНИЕ ОМЭВМ СЕРИИ 1816 ДЛЯ СКПУ

1. Цель работы

Изучение вопросов использования однокристальных микро-ЭВМ серии 1816ВЕ51/31 для построения управляющих устройств узлов коммутации.

2. Литература

1. Гребешков А.Ю. Техника микропроцессорных систем в коммутации: Учебник для вузов. - Самара:ПГУТИ, 2011. - 392 с.

2. Калабеков Б.А. Микропроцессоры и их применение в системах передачи и обработки сигналов.-М.: Радио и связь, 1988.

3. Подготовка к занятию

1.Изучить рекомендованную литературу.

2. Подготовить ответы на контрольные вопросы.

4. Контрольные вопросы

1.Назначение и модификации ОМЭВМ серии 1816ВЕ51/31.

2. Основные технические характеристики ОМЭВМ.

3. Основные функциональные узлы ОМЭВМ.

4. Функционирование ОМЭВМ.

5. Организация памяти ОМЭВМ.

6. Система команд ОМЭВМ.

7. Работа с кросс-Ассемблером ОМЭВМ.

8. Работа с редактором связей ОМЭВМ.

9. Работа с программным эмулятором ОМЭВМ.

5. Порядок выполнения работы

1.Изучить архитектуру ОМЭВМ серии 1816ВЕ51/31 и систему команд.

2.Изучить правила работы с программными средствами отладки программ для ОМЭВМ.

3. Подготовить программу на языке Ассемблер согласно заданию преподавателя.

4. С помощью кросс-ассемблера и редактора связей отладить и перевести программу в машинные коды.

5. С помощью программного эмулятора проверить и показать преподавателю правильность выполнения программы.

6. Составить отчет.

6. Содержание отчета

1. Алгоритм решения поставленной задачи.
2. Программа на языке Ассемблер.
3. Выводы о проделанной работе.

7. Варианты рекомендуемых заданий к лабораторной работе

ЗАДАНИЕ 1

Необходимо считать из порта P1 8-разрядное слово сканирования и определить, в каких разрядах содержится '1'. Номера этих разрядов занести в память в виде массива.

	Вариант 1	Вариант 2	Вариант 3
Начальный адрес массива	20H	40H	60H
Начальный адрес программы	40H	60H	80H

ЗАДАНИЕ 2

Необходимо считать из порта P1 8-разрядное слово сканирования и проанализировать на наличие '1' - по одному разряду из каждой тетрады (тетрада - 4 разряда). При наличии единицы результаты анализа занести в память.

	Вариант 1	Вариант 2	Вариант 3
Анализируемые разряды	3; 7	2; 6	1; 5
Адреса контрольных ячеек	30H	50H	70H
Начальный адрес программы	50H	70H	80H

ЗАДАНИЕ 3

Проверить массив из N ячеек памяти на наличие хотя бы в одной из ячеек числа FFH. При наличии искомого числа вывести в порт P1 - '1

	Вариант 1	Вариант 2	Вариант 3
Количество ЯП в массиве	5	4	6
Начальный адрес массива	30H	40H	50H
Начальный адрес программы	40H	50H	60H

8. Методические указания

8.1 . Архитектура однокристальной микро-ЭВМ 1816ВЕ51/31

Восьмиразрядные однокристальные микро-ЭВМ (ОМЭВМ) 1816ВЕ51/31 предназначены для построения микроконтроллеров и могут быть использованы в составе систем управления технологическим оборудованием.

Серия включает три модификации ОМЭВМ, основное различие между которыми состоит в реализации памяти программ.

КР1816ВЕ51 содержит масочно-программируемое ПЗУ емкостью 4096 байт.

КМ1816ВЕ51 содержит 4096 байт ПЗУ с электрической перезаписью и стиранием ультрафиолетовым излучением.

КР1816ВЕ31 не содержит встроенного ПЗУ, но имеет возможность подключения внешней памяти (ПЗУ иОЗУ).

Основные технические характеристики

1. Встроенная память программ (для модификаций КР1816ВЕ51, КМ1816ВЕ51) емкостью 4096 байт с возможностью расширения до 64 Кбайт за счет внешнего ПЗУ.
2. Встроенная память данных емкостью 128 байт (СОЗУ) с возможностью расширения до 64 Кбайт за счет внешнего ОЗУ.
3. Возможность организации стека теоретически глубиной в128 байт.

4. 32 двунаправленных линии ввода/вывода, организованные в 4 восьмиразрядных порта (с возможностью побайтовой и побитовой адресации).
5. Программируемый последовательный порт ввода/вывода.
6. Два 16-разрядных таймера/счетчика внешних событий.
7. Двухуровневая система прерывания от пяти источников (в том числе от двух внешних источников прерывания).
8. Расширенная система команд, обеспечивающая:
   - прямую побайтовую и побитовую адресацию;
   - двоичную и двоично-десятичную арифметику;
   - индикацию переполнения и определения четности/нечетности.
9. Частота тактового генератора - не более 12 Мгц (длительность одного цикла выполнения команды - не менее 1мкс).
10. ОМЭВМ выполнена по n-МОП технологии в 40-выводном корпусе.
11. Напряжение питания - 5В±5%.

Структурная схема организации логических выводов ОМЭВМ представлена на рис.1.

Назначение выводов

Порт 0 (P0) и порт 2 (P2) составляют мультиплексную шину адреса/данных, предназначенную для подключения внешнего ЗУ (ПЗУ и ОЗУ):

-порт 0 - для ввода/вывода кода данных, а также вывода младшего байта 16-разрядного кода адреса;

-порт 2 - для вывода старшего байта 16-разрядного кода адреса.

Порт 1 (P1) - восьмиразрядный двунаправленный порт ввода/вывода.

Порт 3 (P3) - многофункциональный восьмиразрядный двунаправленный порт ввода/вывода, может использоваться для ввода/вывода информации (аналогично порту 1), а также для ввода/вывода функциональных сигналов следующего назначения:

RXD - вход последовательного порта ввода/вывода;

TXD - выход последовательного порта ввода/вывода;

INT0 - вход внешнего источника прерывания 0;

INT1 - вход внешнего источника прерывания 1;

T0 - вход счетчика событий 0;

T1 - вход счетчика событий 1 (T0,T1 используются для подсчета количества внешних импульсов);

WR - сигнал управления записью данных, принимаемых с мультиплексной шины ввода/вывода;

RD - сигнал управления считыванием данных, выводимых на мультиплексную шину ввода/вывода (WR и RD используются для управления работой внешних ОЗУ и ПЗУ);

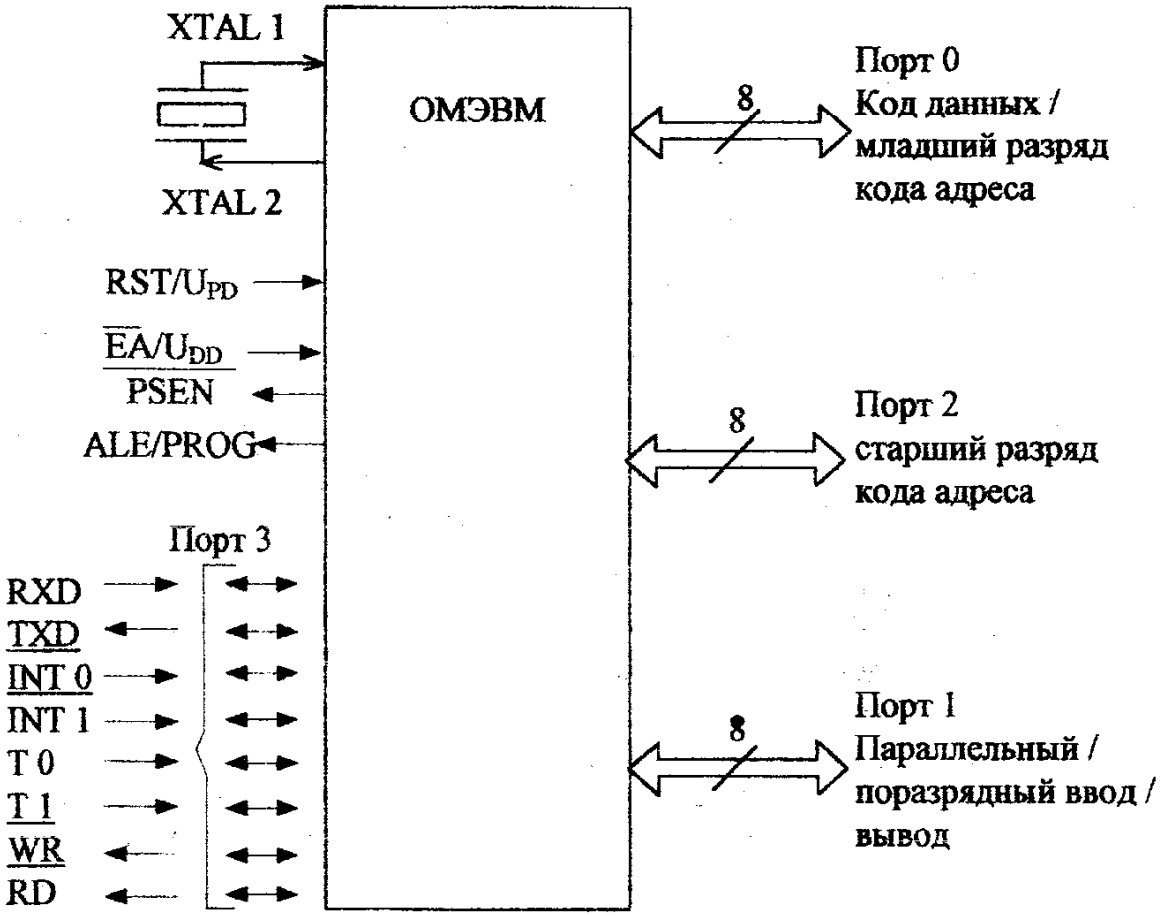

Рис. 1. Назначение выводов.

ALE/PROG - выходной сигнал для фиксации адреса внешней памяти (используется при программировании встроенного ПЗУ, если оно имеется);

PSEN - сигнал, управляющий считыванием из внешнего ПЗУ;

EA/UDD - сигнал, управляющий выборкой из встроенного или внешнего ПЗУ (используется также при программировании встроенного ПЗУ);

RST/UPD - сигнал сброса (установки в исходное состояние регистров ОМЭВМ), используется также при программировании встроенного ПЗУ;

XTAL1, XTAL2 - вход, выход усилителя-генератора синхросигнала (используется при синхронизации от внешнего задающего генератора).

8.2. Функционирование ОМЭВМ

Функциональная схема ОМЭВМ приведена на рис.2. ОМЭВМ включает в себя следующие устройства:

- АЛУ - арифметико-логическое устройство, предназначено для выполнения различных операций (сложение, вычитание, умножение, деление, логические операции и др.);
- Рг1 и Рг2 - регистры временного хранения операндов, подаваемых на входы АЛУ;
- PSW - регистр состояния, содержащий флаги текущего состояния программы:
- C - флаг переноса;
- AC - флаг промежуточного переноса;
- F0 - флаг нуля;
- OV - флаг переполнения;
- P - флаг четности.
- Акк - аккумулятор (регистр А);
- РгВ - регистр,используемый совместно с регистром А (в операциях деления,умножения) или как самостоятельный регистр для хранения данных;
- PC - 16-разрядный счетчик команд (программный счетчик), предназначенный для формирования адреса следующего байта команды (содержимое PC автоматически увеличивается на единицу при выборе очередного байта команды);
- DPTR - 16-разрядный регистр-указатель данных, используется в качестве базового регистра при косвенной адресации (в DPTR предварительно записывается адрес памяти данных, с которыми в последствии производится операция);
- РгА - регистр адреса внешней памяти программ и данных (а также встроенного ПЗУ, если оно имеется);
- Рг команд - предназначен для хранения кода очередной команды ОМЭВМ (код поступает из встроенной или внешней памяти программ);
- SP - регистр-указатель стека, содержит адрес последнего байта, записанного в стек. Аппаратный стек входит в состав внутренней памяти данных (встроенного ОЗУ). Стек используется для передачи параметров между подпрограммами, временного хранения переменных, слова состояния во время программы обслуживания прерываний.

В состав ОМЭВМ входят также:

- два 16-разрядных таймера/счетчика, управление которыми осуществляется программно записью в специальные регистры;

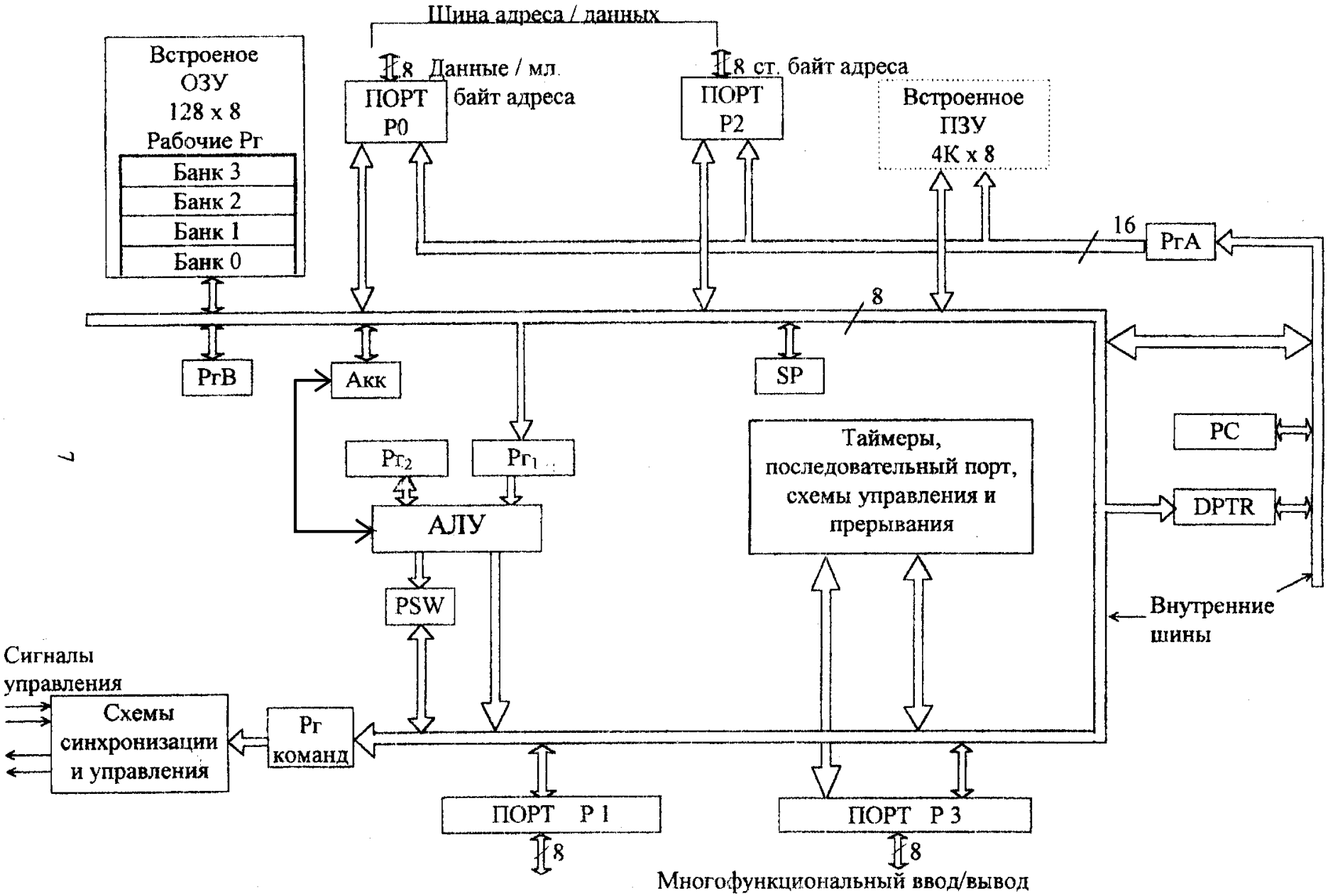

Рис. 2. Функциональная схема

- последовательный порт ввода/вывода. Режимы работы порта (асинхронный, синхронный ввод/вывод, скорость передачи и др.) программируются с помощью специальных регистров. В качестве входа и выхода последовательного порта используются выводы многофункционального порта P3 (см. рис.1).

Другие выводы этого порта используются для:

- организации внешних прерываний ОМЭВМ (INT0 и INT1);
- ввода внешних сигналов для их дальнейшей обработки в ОМЭВМ (T0 и T1).

Схема синхронизации и управления предназначена для дешифрации кода команды и выработки внутренних и внешних сигналов управления.

8.3. Организация памяти ОМЭВМ

Память ОМЭВМ разделяется на память программ (общей емкостью до 64 Кбайт) и память данных (общей емкостью до 64 Кбайт).

Память программ может быть:

- внутренней или встроенной (для КР1816ВЕ51 и КМ1816ВЕ51) емкостью 4096 байт (адреса 0000 - 4095);
- внешней (для КР1816ВЕ31) емкостью до 64 Кбайт (адреса 0000 - 65535);

Память данных разделяется на внутреннюю и внешнюю. Внутренняя и внешняя память имеют единое адресное пространство (адреса 00000 - 65535). Внутренняя память данных включает в себя:

- встроенное ОЗУ емкостью 128 байт (адреса 0000 - 0127);
- регистры специального назначения, расположенные в пространстве адресов 0128 - 0255.

Регистры специального назначения включают все программно доступные регистры (Акк, B, DPTR, PSW, буферные регистры портов P0 - P3, таймеров, а также регистры, управляющие работой основных регистров).

Часть встроенного ОЗУ используется для рабочих регистров, объединенных в 4 банка рабочих регистров по 8 регистров в каждом (адреса с 0000 по 0031 встроенного ОЗУ). Переключение банков производится программно.

8.4. Система команд ОМЭВМ

Система команд ОМЭВМ включает 111 команд, 44% однобайтовых, 41% двухбайтовых и 15% трехбайтовых команд. Обеспечивается побитовая, понибловая (4 бит), побайтовая обработка данных, а также обработка 16-разрядных данных.

Синтаксис большинства команд ассемблерного языка ОМЭВМ состоит из мнемонического обозначения функции, вслед за которым идут операнды, конкретизирующие методы адресации и типы данных. Существует пять способов адресации операндов-источников:

- регистровая адресация;
- прямая адресация;
- косвенно-регистровая адресация;
- непосредственная адресация;
- косвенная адресация по сумме базового и индексного регистра.

Способы адресации, используемые в системе команд ОМЭВМ, приведены в приложении 1. Символы ассемблера ОМЭВМ приведены в приложении 2. Адреса прямоадресуемых регистров приведены в приложении 3.

Сокращенный список команд ОМЭВМ серии 1816ВЕ51/31.

№	Условное обозначение команды на языке Ассемблер	Число байтов	Пояснения	
1	MOV A,Rr (r=0-7)	1	(A) ← (Rr)	Передача содержимого рабочего регистра Rr в A
2	MOV Rr,A (r=0-7)	1	(Rr) ← (A)	Передача содержимого A в рабочий регистр Rr
3	MOV A,direct	2	(A)← (direct)	Передача содержимого прямоадресуемого регистра 'direct' в A
4	MOV direct,A	2	(direct) ← A	Передача содержимого A в прямоадресуемый регистр 'direct'
5	MOV A,#data	2	(A) ← #data	Передача кода данных,указанного во втором байте команды, в A
6	MOV Rr,direct	2	(Rr) ← (direct)	Передача содержимо-

	(r=0-7)			го прямоадресуемого регистра ‘direct’ в рабочий регистр Rr
7	MOV direct,Rr (r=0-7)	2	(direct) ← (Rr)	Передача содержимого рабочего регистра Rr в прямоадресуемый регистр ‘direct’
8	MOV Rr,#data (r=0-7)	2	(Rr) ← #data	Передача кода данных, указанных во втором байте команды, в рабочий регистр Rr
9	MOV direct,direct	3	(direct)←(direct)	Передача содержимого прямоадресуемого регистра-источника (второй байт команды) в прямоадресуемый регистр-приемник (третий байт команды)
10	MOVdirect,#data	3	(direct) ← #data	Передача кода данных,указанного в третьем байте команды, в прямоадресуемый регистр ‘direct’
11	MOV @Rr,A (r=0-1)	1	((Rr))←(A)	Передача содержимого А в регистр, адресуемый регистром-указателем Rr.
12	MOV A,@Rr (r=0-1)	1	(A)←((Rr))	Передача содержимого регистра, адресуемого регистром-указателем Rr, в А
13	ADD A,Rr (r=0-7)	1	(A)←(A)+(Rr)	К содержимому А прибавить содержи-

				мое рабочего регистра Rr. Результат поместить в А
14	ADD A,direct	2	(A)←(A)+(direct)	К содержимому А прибавить байт данных, содержащийся в прямоадресуемом регистре ‘direct’. Результат поместить в А
15	ADD A,#data	2	(A)←(A)+#data	К содержимому А прибавить байт данных. содержащийся во втором байте кода команды. Результат поместить в А
16	SUBB A, Rr (r=0-7)	1	(A)←(A)- (C)-(Rr)	Из содержимого А вычесть содержимое рабочего регистра Rr с учетом заема (содержимого флага переноса). Результат поместить в А
17	SUBB A,direct	2	(A)←(A)-(C)-(direct)	Из содержимого А вычесть байт, содержащийся в прямоадресуемом регистре ‘direct’ с учетом заема (содержимого флага переноса). Результат поместить в А
18	SUBB A,#data	2	(A)←(A)-(C)-#data	Из содержимого А вычесть код данных, содержащийся во втором байте кода команды с учетом заема (содержимого флага

				переноса). Результат поместить в А
19	INC A	1	(A)←(A)+1	Увеличить на 1 содержимое А. Результат поместить в А
20	INC Rr (r=0-7)	1	(Rr)←(Rr)+1	Увеличить на 1 содержимое рабочего регистра Rr. Результат поместить в Rr
21	INC direct	2	(direct)←(direct) +1	Увеличить на 1 содержимое прямоадресуемого регистра 'direct'. Результат поместить в регистр 'direct'
22	DEC A	1	(A)←(A)-1	Уменьшить на 1 содержимое А. Результат поместить в А
23	DEC Rr (r=0-7)	1	(Rr)←(Rr)-1	Уменьшить на 1 содержимое рабочего регистра Rr. Результат поместить в Rr
24	DEC direct	2	(direct)←(direct) -1	Уменьшить на 1 содержимое прямоадресуемого регистра 'direct'. Результат поместить в регистр 'direct'
25	ANL A,Rr (r=0-7)	1	(A)←(A) and (Rr)	Поразрядное логическое 'И'. Результат помещается в А
26	ANL A,direct	2	(A)←(A) and(direct)	Порязрядное логическое 'И' содержимого прямоадресуемого ре-

				гистра ‘direct’ и содержимого А. Результат помещается в А
27	ANL A,#data	2	(A)←(A) and #data	Поразрядное логическое ‘И’ кода данных, указанного во втором байте кода команды, и содержимого А. Результат помещается в А
28	ANL direct,#data	3	(direct)←(direct) and #data	Поразрядное логическое ‘И’ кода данных,указанного во втором байте кода команды, с содержимым прямоадресуемого регистра ‘direct’. Результат помещается в регистр ‘direct’
29	ORL A,Rr (r=0-7)	2	(A)←(A) or(Rr)	Поразрядное логическое ‘ИЛИ’ содержимого рабочего регистра Rr с содержимым А. Результат помещается в А.
30	ORL A,direct	2	(A)←(A)or(direct)	Поразрядное логическое ‘ИЛИ’ содержимого прямоадресуемого регистра ‘direct’и содержимого А. Результат помещается в А.
31	ORL A,#data	2	(A)←(A)or#data	Поразрядное логическое ‘ИЛИ’ кода данных, указанного во втором байте кода команды, с содержимым

				А. Результат помещается в А.
32	CLR A	1	(A)←0	Содержимое всех разрядов А сбрасывается в 0
33	CPL A	1	(A)← not (A)	Инвертирование содержимого А
34	RLC A	1	(An+1)←(An), n=0-6 (Ao)←(C) (C)←(A7)	Циклический сдвиг содержимого А влево на один разряд через разряд флага переноса
35	RRC A	1	(An)←(An+1), n=0-6 (A7)←(C) (C)←(A0)	Циклический сдвиг содержимого А вправо на один разряд через разряд флага переноса
36	LJMP addr 16	3	(PC)←addr 15-0	Безусловный переход по адресу, указанному во втором а(15-8) и первом а(7-0) байтах кода команды. Переход осуществляется загрузкой старшего и младшего байтов PC, соответственно вторым и третьим байтами команды. Адрес перехода может быть любым адресом из 64 Кбайт памяти программ.
37	JZ rel	2	(PC)←(PC)+2, если(A)=0, то (PC)←(PC)+ rel	Если содержимое всех разрядов аккумулятора равно нулю, - переход по адресу, определенному сум-

				мой увеличенного на 2 содержимого PC и кода смещения со знаком. Содержимое A не изменяется.(Если содержимое хотя бы одного разряда A равно 1, - переход к выполнению следующей за JZ команды).
38	JNZ rel	2	(PC)←(PC)+2, если (A)≠0, то (PC)←(PC)+rel	Если хотя бы один разряд аккумулятора равен 1 - переход по адресу, определенному суммой увеличенного на 2 содержимого PC и кода смещения со знаком. Содержимое A не изменяется. (Если содержимое всех разрядов A равно 0, - переход к выполнению следующей за JNZ команды.
39	SETB bit	2	(bit)←1	Установка в '1' прямоадресуемого (вторым байтом кода команды) бита.
40	CLR bit	2	(bit)←0	Сброс в '0' прямоадресуемого (вторым байтом кода команды) бита.
41	JC rel	2	(PC)←(PC)+2; если C=1, то (PC)←(PC)+rel	Если флаг переноса установлен в '1', - переход по адресу, определенному суммой увеличенного на 2 содержимого PC и кода

				смещения со знаком. (Если флаг переноса сброшен в ‘0’, - переход к выполнению следующей за JC команды).
42	JNC rel	2	(PC)←(PC)+2; если(C)=0, то (PC)←(PC)+rel	Если флаг переноса сброшен в ‘0’, - переход по адресу, определяемому суммой увеличенного на 2 содержимого PC и кода смещения со знаком.(Если флаг переноса установлен в ‘1’, - переход к выполнению следующей за JNC команды).
43	B bit,rel	3	(PC)←(PC)+3; если (bit)+1,то (PC)←(PC)+rel	Если прямоадресуемый (вторым байтом кода команды) бит установлен в ‘1’, - переход по адресу, определяемому суммой увеличенного на 3 содержимого PC и кода смещения со знаком. Тестируемый бит не изменяется.(Если прямоадресуемый бит сброшен в ‘0’, - переход к следующей за JB команде).
44	JNB bit,rel	3	(PC)←(PC)+3; если(bit)=0, то (PC)←+rel	Если прямоадресуемый (вторым байтом кода команды) бит сброшен в ‘0’, - переход по адресу, определяемому суммой

				увеличенного на 3 содержимого PC и кода смещения со знаком.Тестируемый бит не изменяется. (Если прямоадресуемый бит установлен в '1', - переход к следующей за JNB команде).

8.5. Правила работы

8.5.1. Запись программы на языке Ассемблер

1. Программа на языке Ассемблер состоит из строк. В каждой строке записывается по одной команде. Записи в строках рекомендуется разбивать на 4 колонки.

Формат строки:

колонка 1	колонка 2	колонка 3	колонка 4
(метка :)	операция	(операнды)	(; коментарии)

Записи в колонках 1, 3, 4 могут отсутствовать.

Имя метки может состоять из любого количества любых символов (строчных и прописных букв в том числе).

2. В начале программы следует задать начальный адрес директивой

ORG (начальный адрес)

В конце программы следует записать директиву

END

3. Числа, фигурирующие в качестве операндов, следует записывать в 16-ричной форме. При этом необходимо поставить символ

- H - после числа или
- $ - перед числом.

Например: $1F или 1FH.

4. Пример записи программы на языке Ассемблер.

```
        ORG     40H
        MOV     A,3FH           ; поместить 3F в Акк
M1:     MOV     B,P1            ; содержимое порта P1 поместить
                                ; в регистр B
        ADD     A,B             ; сложить содержимое Акк и ре-
```

```
                                ;гистра B
JC          M1                  ; если флаг переноса '1', то пере-
                                ; ход на метку M1, иначе - следу-
                                ; ющая команда
MOV         P1,A                ; вывести содержимое Акк в порт
                                ; P1
END
```

5. Имя программы, записанной на ассемблере, должно иметь расширение .asm

Например: ivan.asm

8.5.2. Работа с кросс-ассемблером

Кросс-ассемблер - это пакет программ-трансляторов с языка ассемблера.

Входным файлом является файл программы на языке Ассемблер с расширением .asm.

Выходные файлы имеют следующие расширения:

obj - выходной файл из ассемблера

lst - файл листинга

pak - упакованный выходной файл

Начало работы кросс-ассемблера осуществляется запуском файла

x8051exe

Ассемблер в ответ запросит указание режима вывода:

Listing Destination? (N,T,P,D,E,L,<CR>=N):

где N - печати нет;

T -вывод на терминал;

P- вывод на принтер;

D - запись на диск (текущий);

E - вывод только ошибок;

L - вывод листинга.

По умолчанию - режим N.

Если выбран режим L, то возникает дополнительный запрос устройства вывода листинга:

Listing Destination (T,P,D,<CR>=T):

где T,P,D - те же, что и выше.

Если заказана выдача одних ошибок, то ассемблер спросит:

Error Only Listing Destination (T,P,D,<CR>=T):

Затем программа запросит имя входного файла:

Input filename:

В ответ необходимо ввести имя входного файла с указанием полного пути (расширение .asm при этом можно опустить), например:

a:\ivan

Далее программа запросит имя выходного файла:

Output filename:

В ответ необходимо ввести имя выходного файла или, если оно такое же, как имя входного файла, ответить возвратом каретки ('Enter').

Результат кросс-ассемблирования отобразится на экране дисплея. Для просмотра нажмите 'Ctrl-O', возврат - с помощью тех же клавишь. Указание на ошибки можно посмотреть в листинге программы (файл с расширением .lst).

8.5.3. Работа с редактором связей

Редактор связей предназначен для объединения отдельных частей программы, имеющих перекрестные ссылки. Однако, даже при использовании одного файла программы редактор связей должен быть задействован, так как он формирует выходной файл в нужном формате.

Входным файлом для редактора связей является файл с расширением .obj (сформированный при кросс-ассемблировании).

Выходному файлу необходимо задать расширение .s19.

Начало работы редактора связей осуществляется запуском файла

Link . exe

Программа запросит имя входного файла:

Input Filename:

В ответ необходимо ввести имя входного файла с указанием полного пути (расширение .obj можно опустить), например:

a:\ ivan

Программа выдаст:

Enter Offset For 'CODE':

В ответ нажмите клавишу

'Enter'

Программа запросит имя следующего входного файла. В ответ нажмите клавишу 'Enter'.

Программа запросит имя выходного файла:

Output Filename:

В ответ, если имя выходного файла такое же, как и входного, нажмите

'Enter'

Программа запросит имя библиотечного файла:

Library Filename:

В ответ нажмите клавишу

'Enter'

Программа запросит тип опции (выходного формата):

Options (D,S,A,M,Z,X,H,E,T,1,2,3,<CR>=Default):

В ответ введите **1**

Результат работы редактора связей отобразится на экране дисплея. Для просмотра нажмите 'Ctrl-O'. Если нет ошибок, программа сформирует выходной загрузочный файл с расширением .s19.

8.5.4. Отладка программы на программном эмуляторе

Программный эмулятор avsim51 предназначен для отладки программ в пошаговом, непрерывном ражимах, с остановом в помеченных точках программы.

Вход в эмулятор

Вход в эмулятор осуществляется запуском файла

avsim.exe

Программа предложит выбрать тип ОМЭВМ. В ответ необходимо нажать клавишу 'A'. На экране дисплея возникнет основное поле эмулятора.

Поле эмулятора разбито на 2 части:

- меню команд (нижняя строка экрана);
- окно эмуляции (остальная часть экрана).

Переход от меню команд к окну эмуляции и обратно осуществляется нажатием клавиши 'Esc'.

Меню команд

Меню команд используется для

- загрузки программы в эмулятор;
- задания размеров внешней памяти;
- задания областей памяти, высвечиваемых в окнах эмуляции;
- задания внешних воздействий на эмулятор и др.

Выбор команд в меню осуществляется перемещением курсора влево/вправо. Меню состоит из двух частей. Одна часть высвечена на экране, другая - скрыта. Переход от одной части к другой осуществляется

- с помощью клавиши '↓' или
- последовательным движением курсора влево/вправо.

Каждая команда меню команд имеет структуру вложенных подменю. Выход в основное меню осуществляется с помощью клавишь 'Ctrl-C'.

Загрузка программы в память программ эмулятора

- выбрать команду **'Load'**;
- в подменю выбрать команду **'program'**;
- ввести имя загрузочного файла с расширением .s19, например a:\ivan.s19

Для того, чтобы убедиться в том, что программа загружена, перейти в окно эмуляции ('Esc') и установить начальный адрес программы в счетчике команд PC. В левом окне должна появиться загруженная программа.

Задание адресного пространства памяти данных, высвечиваемых в окнах 'Data Space'

- выбрать команду **'Dump'**;
- в подменю выбрать окно 'Data Spase', в котором вы хотите просматривать содержимое внутренней памяти данных (верхнее окно - '1', нижнее окно - '2');
- в следующем подменю выберите абсолютную адресацию **DUMP: Absolute**;
- наберите начальный адрес интересующей вас области внутренней памяти данных (0000H -00FFH) и введите его.

Проверьте правильность задания области памяти данных по адресам,высвечиваемым в соответствующем окне 'Data Spase'.

Сброс (установка исходного состояния) ОМЭВМ

- выбрать команду **'Reset'**;
- в подменю выбрать команду **'Cpu'.**

При этом произойдет установка всех регистров ОМЭВМ в исходное состояние (например, счетчик команд PC установится в состояние 0000H).

Выход из эмулятора

- выбрать команду **'Quit'**;
- в подменю выбрать **'Exit'** и нажать клавишу 'Enter'.

Окно эмуляции

Окно эмуляции предназначено:

- для отображения содержимого памяти программ (левое окно);

- для отображения и модификации всех регистров и памяти данных ОМЭВМ;
- для задания режимов эмуляции.

Содержимое всех регистров и памяти данных можно менять как перед запуском программы, так и во время выполнения программы на эмуляторе. Для этого необходимо подвести курсор в нужное место и набрать новые данные.

Задавать основные режимы эмуляции можно с помощью следующих клавиш:

- F1 - запуск программы в непрерывном режиме или до точки останова, если она задана;
- F10 - выполнение команд в пошаговом режиме;
- F9 - возврат на одну команду (выполнение предыдущей команды), при этом в окне памяти программ отмечается текущая строка программы;
- F4 - движение курсора вниз по строкам памяти программ с целью последующей фиксации точки останова;
- F2 - движение курсора вверх по строкам памяти программ с целью последующей фиксации точки останова;
- F3 - фиксация точки останова в месте, указанном курсором;
- F5 - установка скорости эмуляции, скорость эмуляции отображается в окне **SPD (медленная, средняя, высокая).**

8.6. Порядок работы

1. Набрать программу на языке Ассемблер в соответствии с п.п.8.5.1. Имя файла программы должно иметь расширение .asm.Для удобства демонстрации программы на эмуляторе рекомендуется в конце программы организовать безусловный переход на ее начало.
2. Отладить программу и создать файлы с расширением .obj и .lst, используя средства кросс-ассемблирования в соответствии с п.п.8.5.2.
3. Сформировать загрузочный файл с расширением .S19, используя средства редактора связей в соответствии с п.п.8.5.3.
4. Отладить и продемонстрировать работу программы с помощью программного эмулятора. Для этого выполнить следующие действия, пользуясь описанием в п.п.8.5.4:

- войти в эмулятор;
- осуществить сброс ОМЭВМ;
- задать начальный адрес массива памяти данных в одном из окон 'Data space' для отображения содержимого интересующих ячеек памяти (в соответствии с заданием);

- установить в счетчике команд PC начальный адрес программы (в соответствии с заданием);
- загрузить программу в память программ, в левом окне должна появиться программа;
- в пошаговом режиме отладить программу, задавая исходные данные
  - для заданий 1 и 2 - на входах порта P1;
  - для задания 3 - заполняя массив памяти данных в одном из окон 'Data Space';
- продемонстрировать работу программы в непрерывном режиме, задавая различные исходные данные.

ПРИЛОЖЕНИЕ 1

СПОСОБЫ АДРЕСАЦИИ, ИСПОЛЬЗУЕМЫЕ В СИСТЕМЕ КОМАНД ОМЭВМ 1816ВЕ51/31

Способ адресации	Адресуемое пространство	Адресуемые регистры
Регистровая	Встроенная память данных	-R7-R0 (рабочие регистры выбранного банка рабочих регистров). -A,B,C (разряд переноса), AB (двухбайтовый регистр). -DPTR (двухбайтовый регистр).
Прямая	Встроенная память данных	-Регистры 0-127(00H-7FH)-встроенное ЗУПВ. -Регистры специального назначения. -128 отдельно адресуемых битов внутри регистров, расположенных по адресам 20H-2FH. -Отдельно адресуемые биты одиннадцати регистров специального назначения.
Косвенно регистровая	Встроенная память данных Внешняя память данных	-Регистры 0-127(00H-7FH)/. В качестве регистров-указателей используются @R1,@R0,@SP(только в командах PUSH и POP). -Младшие ниблы регистров 0-127. В качестве регистров-указателей используются @R0,@R1. -Регистры памяти данных,расширяемой до 64К. В качестве регистров-указателей используются @R1,@R0,@DPTR.
Непосредственная	Память программ	Считывание кодов констант, непосредственно указанных во втором, третьем байтах кода команды.

Через базовый и индексный регистры	Память программ	-Регистры 0-64К. В качестве регистров-указателей используются @DPTR+A и @PC+A. -Просмотр таблиц, зашитых в ПЗУ.

ПРИЛОЖЕНИЕ 2

ОПИСАНИЕ СИМВОЛОВ АССЕМБЛЕРА ОМЭВМ

Обозначе-ние	
A	Аккумулятор
B	Регистр, расположенный в памяти данных. Используется в арифметических операциях.
Rr(r=0-1)	R0,R1 - рагистры-указатели данных.
Rr(r=0-7)	R0-R7 - рабочие регистры текущего банка рабочих регистров.
PSW	Регистр слова состояния программы.
P0	Двунаправленный 8-разрядный порт ввода/вывода.
P1,P2,P3	Квазидвунаправленные 8-разрядные порты ввода/вывода.
C	Флаг переноса.
#data	8-разрядный код константы.
direct	8-разрядный код адреса обращения к прямоадресуемым ячейкам (0-127)
#data16	16-разрядный код константы.
addr16	16-разрядный код адреса назначения. Используется в командах LCAL, LJMP. Обеспечивает обращение и переход в любую точку адресного пространства памяти программ (0-64К)
addr11	11-разрядный код адреса назначения. Используется командами ACALL, AJMP. Обеспечивает обращение и переход внутри той страницы памяти программ объемом 2К, в которой расположен 1-й байт следующей (за выполняемой) команды.
rel	8-разрядный код смещения со знаком. Смещение обеспечивается в диапазоне -128 - +127 относительно первого байта следующей за выполняемой команды. Используется командой SJMP и всеми командами условных переходов.
bit	8-разрядный код адреса прямоадресуемого бита. Используется при обращении к 128 отдельно адресуемым битам ОЗУ и битам регистров специального назначения.
PC	16-разрядный счетчик команд.
DPTR	16-разрядный указатель данных (DPH, DPL).
data(7-0)	Используется для обозначения разрядов (7-0) кода данных в машинном коде.
da(7-0)	direct address (7-0) -используется для обозначения кода адреса прямоадресуемого регистра direct в машинном коде.
bita(7-0)	bit address (7-0) -используется для обозначения кода адреса прямоадресуемого бита в машинном коде.
a(70),a(15-8)	Используется для обозначения кода адреса перехода в машинном коде.

ПРИЛОЖЕНИЕ 3

АППАРАТНЫЕ АДРЕСА РЕГИСТРОВ С ПРЯМОЙ АДРЕСАЦИЕЙ

Регистр	Адрес	Реализуемая функция
P0	80H*	Порт 0
SP	81H	Указатель стека
DPL	82H	Указатель данных (младший байт)
DPH	83H	Указатель данных (старший байт)
TCON	88H*	Регистр управления таймерами
TMOD	89H	Регистр задания режима таймера
TL0	8AH	Младший байт таймера 0
TL1	8BH	Младший байт таймера 1
TH0	8CH	Старший байт таймера 0
TH1	8DH	Старший байт таймера 1
P1	90H*	Порт 1
SCON	98H*	Регистр управления последовательным портом
SBUF	99H	Буфер данных последовательного порта
P2	0A0H*	Порт 2
IE	0A8H*	Регистр разрешения прерываний
P3	0B0H*	Порт 3
IP	0B8H*	Регистр приоритета прерываний
PSW	0D0H*	Слово состояния программы
ACC	0E0H*	Аккумулятор (как прямоадресуемый регистр)
B	0F0H*	B-регистр

* - побитово-адресуемые регистры.