Введение

Основой любой современной вычислительной системы (ВС) являются две составные части: аппаратура и программное обеспечение (Hardware, Software).

В свою очередь, основой программного обеспечения ВС является операционная система (ОС), представляющая собой совокупность специальных программ, предназначенных для управления всеми аппаратными и программными ресурсами ВС, облегчения процесса создания и отладки новых программ, автоматизации их прохождения через вычислительную машину, управления файлами, повышения пропускной способности ВС и производительности труда пользователей.

Операционная система является посредником между ЭВМ и ее пользователями, осуществляя анализ и интерпретацию запросов пользователей и обеспечивая выполнение этих запросов. Конкретная ОС может быть ориентирована на выполнение заданий пользователя в пакетном или диалоговом режиме, а также в режиме реального времени.

Операционная система изучаемой ЭВМ СМ-1800 относится к однопользовательским ОС, содержащим центральный элемент – Монитор и набор программ и данных. Основные характеристики Монитора – предмет рассмотрения в данной лабораторной работе.

Как и предыдущие работы настоящего цикла, данная работа ориентирована на использование эмулятора СМ-1800 – программы, моделирующей работу этой микроЭВМ на современном персональном компьютере.

1 Цель работы

Цель лабораторной работы состоит в изучении центрального элемента инструментальной операционной системы микроЭВМ СМ-1800, которым является программа Монитор.

Изучаются назначение, условия применения, размещение программы Монитор в памяти ЭВМ, а также команды и процедуры, предоставляемые пользователю для подготовки и отладки программ, записываемых в машинном коде.

2 Основные характеристики программы монитор

2.1 Назначение программы

Программа Монитор (MONID) представляет пользователю следующие возможности:

1. Проверять и/или изменять содержимое оперативной памяти (ОП) или регистров общего назначения (РОН);
2. Загружать программы или данные из внешнего запоминающего устройства (ВЗУ);
3. Начинать выполнение программ, находящихся в ОЗУ с использованием или без использования точек прерывания;
4. Распоряжаться набором процедур, позволяющих выполнить ряд стандартных действий при работе с периферийными устройствами.

Первые три возможности обеспечивают диалог пользователь – ЭВМ через системную консоль (терминал ВТА-2000-30). Последняя возможность используется программами пользователей путем обращения к соответствующим процедурам (подпрограммам) с помощью команды CALL.

2.2 Размещение в памяти

Программа Монитор состоит из двух сегментов: резидентного и динамически загружаемого. Носителем резидентного сегмента является ПЗУ, носителем динамически загружаемого – внешнее запоминающее устройство (ВЗУ), реализованного на гибких магнитных дисках (ГМД), с которого он загружается в ОЗУ.

Диаграмма размещения программы Монитор в памяти приведена в Таблице 1.

Таблица 1 - Размещение программы Монитор в памяти

	Адреса	Объем	Назначение	Объем	Назначение
ПЗУ	000H÷003FH	64б	Система обработки прерываний	**2К**	Резидентный сегмент Монитора (собственно программа-Монитор)
	0040H÷0099H	90б	Таблица передач управления функциями Монитора		
	009AH÷07FFH	1894б	Процедуры Монитора		
ОЗУ		992б	Область системного стека	**1К**	Область памяти Монитора, находящаяся в распоряжении пользователя
	0BEH÷0BFFH	32б	Область обработки аппаратных и программных прерываний		
	0C00H÷0FFFH	**1К**	Область памяти, резервируемой для динамически загружаемого сегмента Монитора		

Резидентные сегмент монитора занимает область памяти с адресами от 0000H до 07FFH (2048 байта=2К), в которой:

- ячейки памяти с адресами от 0000H до 003FH (64 байта) резервируются системой прерываний;
- ячейки с адресами 0040H÷0099H (90 байт) содержат таблицу передач управления процедурам монитора (перечень процедур монитора приводится в Приложении 1);
- ячейки памяти с адресами 009A÷07FFH (1894 байта) заняты программами монитора.

Область памяти с адресами 0800H÷0BFFH (1024 байта=1K) находятся в распоряжении программы пользователя во время ее работы:

– ячейки памяти с адресами, меньшими адреса 0BE0H назначаются и используются в качестве системного стека (992 байта);

– ячейки памяти с адресами 0BE0H÷0BFFH (32 байта) резервируются программой монитора для обработки аппаратных и программных прерываний.

Область памяти с адресами 0C00H÷0FFFH (1K) резервируется для размещения второго, динамически загружаемого сегмента монитора. Загрузка этого сегмента выполняется автоматически при первом обращении к процедурам программы монитор. При последующих обращениях повторная загрузка не требуется. Эта область памяти может быть произвольно использована программой пользователя, если в процессе ее работы нет необходимости в обращении к программам монитора.

2.3 Последовательность выполнения работы

Лабораторная работа выполняется на эмуляторе, заменяющем в лабораторных условиях реальную машину СМ-1800.

При загрузке эмулятора и нажатии клавиши «СБРОС» ЭВМ СМ-1800 начинает выполнение программы монитор с адреса 0000H, в результате чего:

1. В указатель стека заносится начальное значение адреса стека – 0BE0H
2. Создается таблица переходов для обработки прерываний в области ОЗУ с адресами 0BE0H÷0BFFH
3. нимается блокировка дверцы накопителя на ГМД (НГМД) *(не эмулируется)*
4. Назначается прозрачный режим обмена с системной консолью, который позволяет вводить/выводить информацию посимвольно с/на консоль без каких-либо ее преобразований;
5. Выдается сообщение о начале работы программы монитор: MONID N, N, где N,N – номер версии;
6. На системную консоль выводится знак (?), следовательно, монитор готов к работе и находится в состоянии ожидания команды.

Далее, при вводе команды работает часть программы монитор, находящаяся в резидентном сегменте (с адреса 0099H по адрес 0800H) и соответствующая введенной команде (ввод команд *L* и *G* в неполном формате заканчивается нажатием клавиши «ВК»).

В случае обращения из пользовательской программы к стандартным процедурам монитора в ОЗУ загружается сегмент

монитора, содержащий программы реализации соответствующих процедур.

Выполнение программы монитор можно прервать нажатием клавиши «ЗБ» (клавиша стирания).

2.4 Команды монитора

В описании команд монитора приняты следующие обозначения:
– все ответы монитора выделены подчеркиванием;
– квадратные скобки используются в общем формате команды для обозначения необязательных параметров.

2.4.1 Команда *L* – «Загрузить программу с дискеты /НГМД/»

Формат команды: *L [V] NAME / [N][BK],*

где:

V – номер устройства, на котором устанавливается дискета (0 или 1), причем, если дискета, с которой загружается программа установлена на устройстве с номером 0, номер устройства можно не указывать;

N – признак передачи управления на начало загружаемой программы (если *N* присутствует, программа будет загружена, но передача управления на начало загруженной программы не произойдет).

Примечание: Эта команда не реализована в эмуляторе.

2.4.2 Команда *D* – «Вывести содержимое области памяти на консоль».

Формат команды: *DXXXX/YYYY,*

где:

XXXX – адрес области памяти, начиная с которого осуществляется вывод;

YYYY – адрес области памяти, до которого выводится ее содержимое.

2.4.3 Команда *S* – «Вывести и изменить содержимое памяти»

Формат команды: *SXXXX YY-,*

где:

XXXX – адрес байта, содержимое которого требуется изменить;

YY – выводимое для сведения пользователя содержимое байта, которое может быть заменено на *ZZ.*

Работа пользователя с этой командой происходит следующим образом. Пользователь, указав адрес байта *(XXXX)*, видит на экране его содержимое *(YY)*. Последовательными нажатием на две клавиши он заменяет содержимое байта двумя шестнадцатеричными цифрами *ZZ.*

После этого на экране высвечивается адрес и содержимое следующего байта по номеру. Если содержимое этого байта не нужно менять, пользователь может нажать либо на клавишу «ВК» и вернуть ЭВМ в монитор, либо на клавишу «Пробел». В последнем случае осуществляется вывод на экран следующего по номеру байта памяти, содержимое которого можно изменить.

2.4.4 Команда *X* – «Вывести содержимое регистров на консоль»

Формат 1: *XX*

По этой команде монитор выводит содержимое всех регистров на консоль.

Формат 2: *XR XX-*,

где:

R – регистр (A, B, C, D, E, H или L), содержимое которого нужно вывести и при желании изменить;

XX – содержимое регистра, которое можно заменить на *ZZ*.

2.4.5 Команда F – «Заполнить память указанным содержимым»

Формат: *FXXXX YYYY ZZ*,

где:

XXXX – адрес, начиная с которого нужно заполнить память;

YYYY – адрес, которым заканчивается заполнение памяти;

ZZ – шестнадцатеричные цифры, которыми требуется заполнить память.

2.4.6 Команда *G* – «Передать управление»

Формат: *G[XXXX] [,YYYY]<ВК>,*

где:

XXXX – адрес программы, которой монитор должен передать управление;

YYYY – адрес точки прерывания программы пользователя (точку прерывания можно использовать только тогда, когда программа находится в ОЗУ).

По команде *G<ВК>* монитор начинает выполнение с последней точки прерывания (или с первого адреса программы, загруженной с дискеты). В этой форме команда *G* чаще всего применяется для возврата в программу, из которой произошло обращение к монитору. Кроме того, она используется для входа в программу, загруженную командой *L* без передачи управления на ее начало.

2.4.7 Команда *M* – «Переслать содержимое области памяти»

Формат: *MXXXX YYYY≥ZZZZ*,

где:

XXXX - адрес, с которого начинается копирование области памяти;

YYYY – адрес, по которому заканчивается копирование области памяти;

ZZZZ – адрес, начиная с которого размещается копия.

2.4.8 Команда *C* – «Сравнить содержимое областей памяти»

Формат*: CXXXX*_*YYYY*≥*ZZZZ,*

где:

XXXX – адрес начала сравниваемой области памяти;

YYYY – адрес конца сравниваемой области памяти;

ZZZZ – адрес начала области памяти, содержимое которой требуется сравнить с содержимым области с адресам *XXXX÷YYYY*.

По ходу сравнения программа выводит на экран адреса и содержимое (в виде шестнадцатеричных цифр) несовпавших байтов двух сравниваемых областей. Выполнение программы может быть в любой момент прервано нажатием любой клавиши на клавиатуре.

2.4.9 Команда *I* – «Разрешить/запретить прерывания»

Команда *I* применяется в двух форматах.

Формат 1: *IE* – «Разрешить прерывание системы»

Формат 2: *ID* – «Запретить прерывание системы»

2.4.10 Команда «ЗБ» - «Отчистить экран»

Нажатие клавиши «ЗБ» приводит к стиранию содержимого экрана, и следующая команда запишется на первой строке экрана.

3 Краткие сведения о машинном языке и языке ассемблера микроЭВМ СМ-1800.

3.1 Формат машинных команд

Для микроЭВМ СМ-1800, использующей 78 основных команд, характерны три формата машинных команд (Таблица 2).

Таблица 2 – Форматы команд

Формат команды	1-й байт	2-й байт	3-й байт
Формат 1	КОП		
Формат 2	КОП	Число	
Формат 3	КОП	Адрес или число	

Формат 1 – однобайтная команда, содержащая лишь код операции (КОП). Операнды в такой команде либо отсутствуют, либо определяются по принципу умолчания, либо находятся в регистрах общего назначения, адреса которых закодированы в этом же байте.

Формат 2 – двухбайтная команда, кроме КОП содержащая восьмиразрядное число, являющееся непосредственным операндом команды или адресом порта ввода/вывода в командах IN и OUT.

Формат 3 – трехбайтная команда, где 2-й и 3-й байты хранят либо 16-разрядное двоичное число, являющееся непосредственным операндом команды, либо 16-разрядный адрес операнда. В обоих случаях второй байт (т.е. байт, следующий за байтом с кодом операции) хранит младшую часть непосредственного операнда или адреса, а третий байт – старшую часть. Эту особенность следует учитывать программисту при интерпретации дампов памяти (отображений кодов в шестнадцатеричной системе исчисления микроЭВМ СМ-1800).

Если второй и третий байты команды содержат непосредственный операнд, который подлежит занесению в пару регистров, то в результате выполнения команды содержимое 2-го байта будет помещено в младший регистр пары, а содержимое 3-го – в старший регистр.

Машинно-ориентированный язык символического кодирования программ – ассемблер – предполагает замену цифрового кода операции машинной команды на последовательность латинских букв. Обычно эта последовательность является аббревиатурой английских слов, описывающих данную операцию. Например, LXI – Load Extended Immediate (загрузить двойной (длинный) непосредственными...).

Язык ассемблера допускает введение символических обозначений адресов ячеек памяти – идентификаторов, что облегчает работу программиста.

3.2 Интерпретация дампа на язык ассемблера

Требуется интерпретировать на язык ассемблера следующий дамп: 21 20 40 CD 55 00 FE 30 CA 10 40 77 23 C3 03 40 AF 77 01 20 40 CD 4F 00 C3 40 00.

Первая пара шестнадцатеричных цифр в приведенном дампе – 21. Это код команды LXI H (см. Приложение 2), осуществляющей непосредственную загрузку содержащегося в ней операнда в пару регистров H и L. Команда LXI – трехбайтная, следовательно, следующие две пары цифр являются операндом команды. Содержимое третьего байта команды (40) загружается в первый регистр заданной пары, т.е. в регистр H, а содержимое второго байта команды (20) в регистр L – второй регистр заданной пары. Таким образом, ассемблерный текст этой команды выглядит так: LXI H, 4020 H, где буква H после кода 4020 говорит об использовании при записи операнда

шестнадцатеричной системы исчисления. Отсутствие буквы H заставит транслятор рассматривать данный код как десятичное число при переводе его (во время трансляции) в двоичную систему.

Следующая в рассматриваемом дампе пара цифр – CD, являющаяся кодом команды CALL, обеспечивающей обращение к подпрограмме. Следовательно, последующие два байта дампа интерпретируются как адрес начала подпрограммы (точки входа) 0055 H.

Код FE соответствует команде сравнения с непосредственным операндом CPI. Второй операнд этой двухбайтной команды находится в следующем байте дампа (30). Сравнение этого кода производится с содержимым аккумулятора (первый операнд). Результатом выполнения команды является установление всех флагов в зависимости от величин сравнимых чисел, как это происходит при вычитании.

Следующая пара цифр CA определяет код команды JZ - условный переход по адресу в случае, если значение флага Z = 1. Адрес перехода составляется третьим и вторым байтами команд (4010H).

Коду 77 соответствует однобайтная команда пересылки байта – MOV. В ее коде зашифрованы адреса источника и приемники данных. В данном случае источником является аккумулятор, приемником – ячейка ОЗУ, косвенный адрес находится в паре регистров H и L.

Код операции 23 принадлежит команде INX H, которая увеличивает содержимое сдвоенного регистра H,L на единицу, тем самым продвигая косвенный адрес к следующему байту.

Трехбайтная команда безусловного перехода JMP имеет код операции C3. Адрес перехода составляется из ее 3-го и 20-го байтов (4003H).

Пара цифр AF определяет логическую команду исключающего ИЛИ (или сложение по модулю 2). Команда XRA A определяет как первый, так и второй операнд в аккумуляторе. Это приводит к тому, что ее выполнение всегда обнуляет аккумулятор.

Далее известный код 77 команды MOV M, A.

Код 01 принадлежит команде LXI B, которая загружает непосредственно содержимым своего третьего и второго байтов (40 и 20) регистры B и C соответственно.

Код CD определяет команду вызова подпрограммы по адресу 004FH. Программа завершается командой обращения к процедуре монитора путем безусловного перехода по адресу 0040H.

Итог интерпретации кодов дампа памяти – программа на языке ассемблера:

```
LXI     H, 4020H
CALL    55H
CPI     30H
JZ      4010H
MOV     M, A
INX     H
JMP     4003H
XRA     A
MOV     M, A
LXI     B, 4020H
CALL    4FH
JMP     40H
```

3.3 Анализ программы

LXI H, 4020H – в пару регистров H,L загружается адрес памяти 4020H для работы процедуры ввода TTIO.

CALL 55H – вызывается подпрограмма монитора TTIO – ввод символа в аккумулятор с эхом на консоль.

CPI 30H – введенный по процедуре TTIO в аккумулятор символ, сравнивается с кодом 30H.

JZ 4010H – переход, если флаг Z установлен, причем флаг нуля устанавливается после работы команды CPI только в том случае, когда с консоли процедурой TTIO будет введен символ нуля. В противном случае, когда перехода по адресу 4010H не произойдет, и будет выполняться команда, следующая за командой JZ.

MOV M, A – пересылка введенного TTIO символа по адресу, содержащемуся в регистрах H, L.

INX H – инкремент пары регистров H, L, т.е. увеличение значения адреса памяти на единицу.

JMP 4003H – безусловный переход по адресу 4003H.

XRA A – обнуление аккумулятора.

MOV M, A – пересылка нулевого содержимого аккумулятора в память по адресу, содержащемуся в регистра H, L, т.е. завершение текста нулевым (пустым) байтом. Это требование процедуры TTCLF для обнаружения конца выводимого текста.

LXI B, 4020H – загрузка в пару регистров B, C начального адреса, по которому записан символьный текст для процедуры вывода TTCLF.

JMP 40H – выход в Монитор в режиме ожидания ввода команды Монитора.

4 Последовательность изучения программы Монитор

Выполнение данной лабораторной работы осуществляется с помощью эмулируемого видеотерминала, расположенного на вкладке «Монитор».

Нажатие на клавишу «СБРОС», расположенной на ПКУ (можно воспользоваться функциональной клавишей F2 клавиатуры ПК), вызывает печать имени монитора и признака готовности к приему команды:

MONID 1.0

?

Ожидая ввода команды с консоли, процессор ЭВМ не стоит, а непрерывно выполняет программу циклического опроса портов ввода, связанных с клавиатурой. Набираемые при вводе команды символы наполняют буфер ограниченного размера. К анализу (реализации) набранной команды монитор приступает или после того, как получит код клавиши «ВК» («ENTER»), или, если буфер оказывается полностью заполненным.

Работа с программой монитор осуществляется в следующей последовательности:

1. Запустить программу эмулятора, перейти на вкладку "Монитор". Нажать F2 или выбрать из главного меню Команды > Сброс.
2. Вывести на экран начальный фрагмент кодов программы монитор, начиная с адреса 0000H по адрес 0100H. Определить значения первых четырех команд, начиная с адреса 0000H, пользуясь приложением 2.
3. Создать копию монитора в ОЗУ по адресу 4000H.

4. Сравнить копию с оригиналом и проанализировать результат (несовпадение байтов).
5. Заполнить всю память, например, единицами(F 0000/FFFF 11).
6. Сравнить содержимое области памяти от адреса 0C00H до адреса 0FFFH с содержимым в памяти, начиная с адреса 8000H, а также от адреса 0000H до адреса 0FFFH с содержимым памяти, начиная с адреса 8000H. Проанализировать результаты.
7. Записать в ОЗУ программу ввода текстовой константы, начиная с адреса 4000H в машинных кодах:

Адрес	Машинный код	Код Ассемблера
4000	21 20 40	LXI H, 4020H
4003	CD 55 00	CALL 55H
4006	FE 30	CPI 30H
4008	CA 10 40	JZ 4010H
400B	77	MOV M, A
400C	23	INX H
400D	C3 03 40	JMP 4003H
4010	AF	XRA A
4011	77	MOV M, A
4012	01 20 40	LXI B, 4020H
4015	CD 4F 00	CALL 4FH
4018	C3 40 00	JMP 40H

8. Проверить правильность набора программы, высветив на экране дамп памяти 4000H÷4100H. Исправить возможные ошибки.
9. Передать управление введенной программе без задания точек программного прерывания. С клавиатуры консоли ввести произвольный текст, закончив его символом *0*. Убедиться в правильности введенного текста по распечатке на экране.
10. Ввести следующие текстовые константы:

1-е_слагаемое _?0

2-е_слагаемое _?0

Сумма=0, начиная с адресов соответственно: 4020H, 4044H, 4058H.

Для изменения адреса записываемой константы изменить содержимое 2-го байта в кодах команд LXI. Для повторного использования программы передать управление на начало командой монитора *G*.

Внимание! Программа не защищена от ошибок ввода, т.е. исправление «опечатки» движением курсора (BACK SPACE) на

экране ведет к удлинению текста, который с нажатием очередной клавиши продолжает накапливаться в последующих байтах. Это может привести к нежелательному «наложению» областей текстов и программ.

11. Записать в ОЗУ программу сложения однобайтных чисел в машинных кодах, начиная с адреса 4060H:

Адрес	Машинный код
4060	CD 49 00
4063	01 20 40
4066	CD 4C 00
4069	CD 67 00
406C	57
400D	CD 49 00
4070	01 44 40
4073	CD 4C 00
4076	CD 67 00
4079	82
407A	CD 49 00
407D	01 58 40
4080	CD 4C 00
4083	CD 61 00
4086	CD 49 00
4089	C3 60 40

Проверить правильность записи программы. Выполнить программу, вводя двузначные шестнадцатеричные слагаемые. Определить действия, выполняемые по каждой команде. Произвести сложение следующих чисел: 87+78, 88+78, 88+79.

Для окончания работы программы нажать клавишу СБРОС на ПКУ или выбрать из главного меню эмулятора: Команды > Сброс.

12. Повторно запустить программу с установкой контрольной точки программного прерывания по адресу 407AH. Ввести слагаемые, проверить содержимое регистров A и D. Запустить программу на дальнейшее выполнение командой G.
13. Перейти на вкладку «Пульт контроля и управления». После нажатия на клавишу «Сброс» установить на ПКУ режим «Остановка по адресу». На регистре «Адрес» набрать адрес 0030h. Запустить повторно программу сложения двух чисел с программным прерыванием по адресу 407Ah. Ввести слагаемые и убедиться с помощью ПКУ, что процессор остановился по адресу 0030h. Нажать клавишу «Сброс». Прочитать содержимое байта 407Ah. Сравнить с текстом программы. Установить, пользуясь Приложением 2, значение команды, оказавшейся по этому адресу.
14. Выключить эмулятор СМ1800.

Обсуждение этапов.

Перед тем, как приступить к практическому выполнению работы на эмуляторе, необходимо подготовить список команд монитора, которые будут использоваться на всех этапах работы.

Так, для выполнения этапа 6 необходимо дважды применить команду CXXX/YYYY>ZZZZ, а для выполнения этапа 7 – команду SXXXX. Выполняя этап 7 с помощью команды S, нужно учесть, что замена содержимого байтов происходит по последовательным адресам. Дойдя до правого края экрана, нужно закончить выполнение команды S нажатием клавиши «ВК». На следующей строке появится номер очередного, еще не исправленного байта и его содержимое. Эта информация полезна для того, чтобы продолжать занесение в ОЗУ текста программы именно с этого адреса, заново набирая команду S.

На этапе 10 два раза выполняется одна и та же последовательность действий: дважды используется команда S для исправления адресов вводимых текстов, запускается программа командой *G* и выводится во время выполнения программы один из трех предлагаемых текстов.

Выполнение этапа 12 требует подробного предварительного анализа программы сложения двух чисел с представлением ее команд на языке ассемблера.

Наибольшую сложность представляет для понимания этапа 13. Его выполнение отличается от выполнения этапа 12 применением ПКУ для физической остановки процессора по адресу 0030H. В пункте 12 после приостановки по адресу 407AH программа успешно продолжает выполняться, если набрать команду *G* без параметров. Никакой «порчи» текста программы не происходит. При выполнении же этапа 13 после остановки по адресу 0030H в ячейке 407AH обнаруживается вместо набранного ранее кода CD код F7. Это код команды перехода по адресу 0030H. Оказывается, команда монитора *G* скрывает за собой выполнение процедуры, по которой при наличии требования об остановке по точке программного прерывания в этот адрес записывается код F7. Однако предварительно код исходной программы, стоявший по этому адресу, запоминается в стеке. После обработки прерывания по адресу 0030H содержимое ячейки 407AH будет восстановлено. Вот почему на этапе 12 не происходит никакой «порчи» программы при ее дальнейшей работе. На этапе 13 нарушена (нажатием клавиши «СБРОС») последовательность обработки программного прерывания. Таким образом, вскрывается механизм работы процедуры *G* по организации прерывания.

5 Содержание работы

5.1 Задание

Изучить назначение, правила использования, размещение в ОЗУ программы Монитор.

На примере пользовательской программы, представленной в кодах и на языке ассемблера, получить навыки расшифровки дампов в ОЗУ, анализа содержания программы и работы с программой посредством использования команд и процедур монитора.

5.2 Порядок выполнения работы

1. Изучить методические указания.
2. Подготовить ответы на контрольные вопросы.
3. Выявить команды монитора, с помощью которых выполняются отдельные этапы лабораторной работы.
4. Проанализировать программу сложения однобайтных чисел, восстановив ее текст на языке ассемблера.
5. Выполнить все этапы работы.

6 Оформление результатов работы

Отчет о выполненной работе должен содержать:

- запись использованных команд монитора и результатов их выполнения, зафиксированных в ходе выполнения работы, на каждом этапе работы;
- текст программы сложения двух двухзначных шестнадцатеричных чисел на ассемблере с подробным комментарием;
- упрощенную блок-схему работы программы команды монитора *G* на основе выполнения этапа 13.

7 Контрольные вопросы

1. Какие возможности представляет пользователю программа Монитор?
2. Сколько килобайт ПЗУ занимает программа Монитор? Какова величина динамически загружаемого сегмента Монитора и для чего он предназначен?
3. Что такое системный стек?

4. Какие действия производит Монитор после нажатия клавиши «СБРОС» (или F2 клавиатуры ПК)?
5. Приведите примеры основных команд монитора.
6. В каких форматах хранится в ОЗУ команды микроЭВМ СМ-1800?
7. Какой команде языка ассемблера соответствует машинный код 01 58 40?
8. Объясните, чем отличается машинное представление в памяти ЭВМ СМ-1800 числа 0 от представления символа 0?
9. Объясните подробно (по заданию преподавателя) один из этапов выполнения лабораторной работы.

Литература

1. Видеотерминал и клавиатура микроЭВМ: Методические указания к лабораторной работе.- СПб.: СПбГТИ(ТУ), 2006.-20с.
2. МикроЭВМ СМ-1800 и её эмулятор на ПК: Методические указания к лабораторной работе.- СПб.: СПбГТИ(ТУ), 2006.-21с.
3. Обработка прерываний.Модуль таймера: Методические указания к лабораторной работе.- СПб.: СПбГТИ(ТУ), 2006.-19с.
4. Программирование в кодах для микроЭВМ СМ-1800: Методические указания.- СПб.:СПбГТИ(ТУ), 2006.-24с.
5. Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем: Учебник для вузов. –СПб.:Питер, 2004. -668с.

Приложение 1

Перечень процедур монитора, доступных прикладным программам

MON – вход в режим ожидания ввода команды монитора (40H)
MONER – вход в монитор с идентификацией ошибки в программе (43H)
TTSTAT – проверка статуса ввода и ввод символа с консоли (46H)
TTI – ввод кода символа в A без эха на консоль (52H)
TTIO – ввод кода символа в A с эхом на консоль (55H)
INHEX – ввод двух шестнадцатеричных цифр в A (67H)
INHX2 – ввод четырех шестнадцатеричных цифр в регистры B,C (6AH)
TTO – вывод символа на экран из аккумулятора (58H)
CRLF – переход на новую строку (49H)
TTCON – вывод строки текста, начиная с адреса, записанного в B,C до нулевого байта (4CH)
TTCLF – вывод строки текста с переходом на новую строку, начиная с адреса в паре B,C до нулевого байта (4FH)
OUTHX – вывод двух шестнадцатеричных цифр из A (61H)
OUTH2 – вывод четырех шестнадцатеричных цифр из B,C (64H)
DREG – вывод содержимого регистров (58H)
DMEM – шестнадцатеричный дамп памяти, начиная с адреса в паре B,C и заканчивая адресом из пары H,L (5EH)
LPO – вывод символа на печатающее устройство (97H)
NIBBLE – проверка и перевод шестнадцатеричной цифры
MOVE – перемещение области памяти (8EH)
COMP – сравнение областей памяти (91H)
FILL – заполнение памяти указанным содержимым (94H)
BDO – открытие дискеты (70H)
BRDR – чтение сектора с дискеты в буфер (73H)
BWDR – запись сектора из буфера на дискету (76H)
BDDR - запись признака стертых данных (79H)
BDC – закрытие дискеты (7CH)
BDD – опускание считывающей головки (7FH)
BLDR – чтение сектора с дискеты (82H)
BUDR – запись сектора на дискету (85H)
LOAD – загрузка программы с дискеты (6DH)
LINK – загрузка и запуск программы (88H)

Примечание:
В скобках указаны адреса точек входа в процедуры. Подробное описание работы процедур Монитора приводится в п.3.

Приложение 2

Коды операций команд в порядке возрастания

00 NOP	40 MOV B,B	80 ADD B	C0 RNZ
01 LXI B,XXXX	41 MOV B,C	81 ADD C	C1 POP B
02 STAX B	42 MOV B,D	82 ADD D	C2 JNZ XXXX
03 INX B	43 MOV B,E	83 ADD E	C3 JMP XXXX
04 INR B	44 MOV B,H	84 ADD H	C4 CNZ XXXX
05 DCR B	45 MOV B,L	85 ADD L	C5 PUSH B
06 MVI B,XX	46 MOV B,M	86 ADD M	C6 ADI XX
07 RLC	47 MOV B,A	87 ADD A	C7 RST 0
08 - - - -	48 MOV C,B	88 ADC B	C8 RZ
09 DAD B	49 MOV C,C	89 ADC C	C9 RET
0A LDAX B	4A MOV C,D	8A ADC D	CA JZ XXXX
0B DCX B	4B MOV C,E	8B ADC E	CB - - - -
0C INR C	4C MOV C,H	8C ADC H	CC CZ XXXX
0D DCR C	4D MOV C,L	8D ADC L	CD CALL XXXX
0E MVI C,XX	4E MOV C,M	8E ADC M	CE ACI XX
0F RRC	4F MOV C,A	8F ADC A	CF RST 1
10 - - - -	50 MOV D,B	90 SUB B	D0 RNC
11 LXI D,XXXX	51 MOV D,C	91 SUB C	D1 POP D
12 STAX D	52 MOV D,D	92 SUB D	D2 JNC XXXX
13 INX D	53 MOV D,E	93 SUB E	D3 OUT XXXX
14 INR D	54 MOV D,H	94 SUB H	D4 CNC XXXX
15 DCR D	55 MOV D,L	95 SUB L	D5 PUSH D
16 MVI D,XX	56 MOV D,M	96 SUB M	D6 ACI XX
17 RAL	57 MOV D,A	97 SUB A	D7 RST 2
18 - - - -	58 MOV E,B	98 SBB B	D8 RC
19 DAD D	59 MOV E,C	99 SBB C	D9 - - - -
1A LDAX D	5A MOV E,D	9A SBB D	DA JC XXXX
1B DCX D	5B MOV E,E	9B SBB E	DB IN XX
1C INR E	5C MOV E,H	9C SBB H	DC CC XXXX
1D DCR E	5D MOV E,L	9D SBB L	DD - - - -
1E MVI E,XX	5E MOV E,M	9E SBB M	DE SBI XX
1F RAR	5F MOV E,A	9F SBB A	DF RST 3
20 - - - -	60 MOV H,B	A0 ANA B	E0 RPO
21 LXI H,XXXX	61 MOV H,C	A1 ANA C	E1 POP H
22 SHLD XXXX	62 MOV H,D	A2 ANA D	E2 JPO XXXX
23 INX H	63 MOV H,E	A3 ANA E	E3 XTHL
24 INR H	64 MOV H,H	A4 ANA H	E4 CPO XXXX
25 DCR H	65 MOV H,L	A5 ANA L	E5 PUSH H
26 MVI H,XX	66 MOV H,M	A6 ANA M	E6 ANI XX

27 DAA	67 MOV H,A	A7 ANA A	E7 RST 4
28 - - - -	68 MOV L,B	A8 XRA B	E8 RPE
29 DAD H	69 MOV L,C	A9 XRA C	E9 PCHL
2A LHLD XXXX	6A MOV L,D	AA XRA D	EA JPE XXXX
2B DCX H	6B MOV L,E	AB XRA E	EB XCHG
2C INR L	6C MOV L,H	AC XRA H	EC CPE XXXX
2D DCR L	6D MOV L,L	AD XRA L	ED - - - -
2E MVI L,XX	6E MOV L,M	AE XRA M	EE XRI XX
2F CMA	6F MOV L,A	AF XRA A	EF RST 5
30 - - - -	70 MOV M,B	B0 ORA B	F0 RP
31 LXI SP,XXXX	71 MOV M,C	B1 ORA C	F1 POP PSW
32 STA XXXX	72 MOV M,D	B2 ORA D	F2 JP XXXX
33 INX SP	73 MOV M,E	B3 ORA E	F3 DI
34 INR M	74 MOV M,H	B4 ORA H	F4 CP XXXX
35 DCR M	75 MOV M,L	B5 ORA L	F5 PUSH PSW
36 MVI M,XX	76 HLT	B6 ORA M	F6 ORI XX
37 STC	77 MOV M,A	B7 ORA A	F7 RST 6
38 - - - -	78 MOV A,B	B8 CMP B	F8 RM
39 DAD SP	79 MOV A,C	B9 CMP C	F9 SPHL
3A LHLD XXXX	7A MOV A,D	BA CMP D	FA JM XXXX
3B DCX SP	7B MOV A,E	BB CMP E	FB EI
3C INR A	7C MOV A,H	BC CMP H	FC CM XXXX
3D DCR A	7D MOV A,L	BD CMP L	FD - - - -
3E MVI A,XX	7E MOV A,M	BE CMP M	FE CPI XX
3F CMC	7F MOV A,A	BF CMP A	FF RST 7

Условные обозначения:
XXXX – адрес ячейки памяти или данные
XX – адрес порта или данные
- - - - – код операции не используется

Содержание