ТВЕРСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра автоматизации технологических процессов

АППАРАТНЫЕ И ПРОГРАММНЫЕ СРЕДСТВА IBM-СОВМЕСТИМЫХ КОМПЬЮТЕРОВ

Методические указания к лабораторным работам по курсу "ЭВМ в системах управления" для студентов специальности 21.03

Тверь 1998

Настоящее пособие предназначено для закрепления теоретического курса "ЭВМ в системах управления" студентами специальности "Автоматизация технологических процессов и производств". Рассмотрены вопросы разработки и отладки программ на языке Ассемблера для IBM-совместимых компьютеров, управление ресурсами вычислительной системы и внешними устройствами.

Пособие может быть полезно для студентов других специальностей, связанных с работой на персональных ЭВМ.

Методические указания обсуждены и рекомендованы к печати на заседании кафедры АТП (протокол № 9 от 4.02.1998 г.).

Составители: А.Л.Калабин, Г.А. Дмитриев, А.А. Шейнман

С О Д Е Р Ж А Н И Е

ЛАБОРАТОРНАЯ РАБОТА № 1

ВЗАИМОДЕЙСТВИЕ ПРОГРАММИСТА С ОПЕРАЦИОННОЙ СИСТЕМОЙ ПРИ СОЗДАНИИ ПРОГРАММЫ НА ЯЗЫКЕ АССЕМБЛЕРА

В настоящей работе рассматривается последовательность выполнения необходимых этапов от начала разработки программы на языке Ассемблера до ее отладки. Напомним, что основными этапами разработки программы являются:

- получение исходного файла (ввод текста программы в компьютер и редактирование текста);
- получение объектного файла (трансляция с языка Ассемблера);
- получение загрузочного файла;
- отладка программы;
- исполнение отлаженной программы.

Каждый из перечисленных этапов выполняется под управлением дисковой операционной системы MS DOS.

Для создания исходного файла применяется программа, называемая Редактором. Редактор позволяет вводить нужный текст и изменять его по мере надобности. Выбор конкретного Редактора целиком возлагается на пользователя в зависимости от его опыта и навыков работы.

Файл, созданный Редактором, должен иметь расширение .ASM .

1. Запуск программы Макроассемблера

После того, как создан исходный файл, содержимое которого записано на диске в коде ASCII, применяется программа Макроассемблера MASM, транслирующая исходный файл в объектный файл, в котором исходная программа записана на машинном языке (в кодах команд микропроцессора К8086).

Перед запуском Макроассемблера следует убедиться, что на диске имеется программа с именем MASM.EXE. После загрузки программы MASM на экран дисплея выдается сообщение

Source filename [.ASM]:

(Исходный файл)

Предположим, что при создании исходного файла ему было присвоено имя SORT.ASM, тогда после двоеточия указанного выше сообщения вводится имя файла SORT (расширение .ASM можно опустить) и нажимается клавиша ENTER. На экран выводится следующее сообщение:

Object filename [SORT.OBJ]:
(Объектный файл)

Это сообщение запрашивает имя объектного файла, которому должно соответствовать расширение .OBJ. После присвоения имени нажимается клавиша ENTER. Если Вы хотите, чтобы объектный файл имел то же имя, что и исходный, то после двоеточия ничего не вводится, а просто нажимается клавиша ENTER.

Source listing [NUL.LST]:
(Файл листинга)

Здесь также следует присвоить имя файлу, содержащему листинг - полный список адресов ячеек памяти и записанных в них кодов команд или данных с соответствующим текстом исходного файла. Если листинг не нужен, то нажимается клавиша ENTER. Если создается файл листинга, то он получит расширение .LST. Последним сообщением будет:

Cross-reference [NUL.CRF]
(Файл ссылок)

Файл перекрестных ссылок содержит таблицу всех символических имен, определенных в программе, причем каждому имени сопоставляется номер строки исходной программы, в которой появляется это имя. Если файлу перекрестных ссылок присваивается имя, то он создается, иначе нет. После последнего нажатия клавиши ENTER начнется трансляция. Если во время трансляции обнаружатся синтаксические ошибки, то информация о них выводится на экран дисплея и в файл листинга.

При успешном завершении работы программы MASM на экране появляется сообщение:

0 Warrning Errors
0 Sever Errors.

В случае создания файла листинга его можно вывести на экран с помощью системной программы TYPE или нажав клавишу F3, если Вы работаете в Norton'e.

1. Получение загрузочного файла

Следующий этап разработки программы заключается в формировании загрузочного модуля с помощью системной программы редактора связей LINK. Исходным файлом для этого этапа служит объектный файл с расширением .OBJ .
Редактор связей соединяет отдельно созданные объектные модули, генерирует выполняемый загрузочный модуль, разрешает внешние перекрестные ссылки, ищет файлы в библиотеке, создает листинг карты связей. В таблице приведены типы файлов, с которыми работает программа LINK.

Таблица.

Входные и выходные файлы программы

Тип файла	Назначение файла	Расширение файла	Чем создается	Кем/чем используется
Входной Входной Выходной Выходной	Объектный Библиотечный Листинг Загрузочный модуль	.OBJ .LIB .MAP .EXE	Макроассемблером Компилятором Программой LINK Программой LINK	Программой Программой Пользователем Перемещающим загрузчиком

Работа с редактором связей производится в следующем порядке. Сначала загружается программа LINK, после успешной загрузки которой должно появиться первое сообщение:

Object Modules [.OBJ]:
(Объектные модули)

В ответ на это сообщение после двоеточия необходимо ввести имя объектного файла с расширением .OBJ (расширение указывать не обязательно). Если число связываемых файлов больше, чем один, то их имена вводятся через пробел или разделитель + (знак плюс). Например,

Object Moduls [.OBJ]: SORT + TROS

файлы связываются в том же порядке, в каком их имена передаются программе редактора связей. Первым указывается файл, в котором записан главный модуль. Для других файлов порядок указывания безразличен.

После нажатия клавиши ENTER должно появиться следующее сообщение (для примера связывания двух файлов с именами SORT и TROS):

Run File [SORT.EXE]: (Исполняемый файл)

После двоеточия вводится имя выполняемого файла, который получает расширение .EXE . Если в данную строку ничего не вводится, то именем файла становится первое из имен, указанных в предыдущем сообщении. Далее после нажатия клавиши ENTER выдается сообщение:

List File [NUL.MAP]:
(Файл листинга)

файл листинга не создается, если после двоеточия нажать клавишу ENTER, но если до нажатия на клавишу ввести имя файла, то будет создан файл карты связей.
Последний запрос программы LINK касается правила ввода файла из библиотеки программ. В этом случае в ответ на сообщение

Libraris [.LIB]:
(Библиотеки)

требуется ввести обозначение устройства, где размещена библиотека и имя файла с расширением .LIB. Если просто нажимается клавиша ENTER, то просмотра библиотеки не будет.

3. Порядок выполнения работы

1. Изучите команды Ассемблера, входящие в программы 1 и 2
2. Изучите, прокомментируйте и сравните две программы

Программа 1

```
SEG1  SEGMENT
      OPER1 DB 5
      OPER2 DB 3
SEG1  ENDS
SEG2  SEGMENT
      RESULT DB ?
SEG2  ENDS
CODE        SEGMENT
      ASSUME CS:CODE, DS:SEG1
START:            MOV AX,SEG1
            MOV DS,AX
            MOV AH,OPER1
            ADD AH,OPER2
            ASSUME DS:NOTHING, DS:SEG2
            MOV AX,SEG2
            MOV DS,AX
            MOV RESULT,AH
            MOV AH,4CH
            INT 21H
CODE ENDS
      END START
```

Программа 2

```
SEG1  SEGMENT
      OPER1 DB 5
      OPER2 DB 3
SEG1   ENDS
SEG2  SEGMENT
      RESULT DB ?
SEG2 ENDS

CODE        SEGMENT
      ASSUME CS:  CODE,  DS:SEG1,  ES:SEG2
START:            MOV AX, SEG1
            MOV  DS, AX
```

```
        MOV  AX, SEG2
        MOV  ES, AX
        MOV  BH, OPER1
        ADD  BH, OPER2
        MOV  RESULT, BH
        MOV AH, 4CH
        INT 21H
CODE    ENDS
        END START
```

3. Выберите более эффективную программу и получите загрузочный модуль в соответствии с пунктами 1 и 2.
4. Сформулируйте критерии оценки эффективности программ.

ЛАБОРАТОРНАЯ РАБОТА № 2

ОТЛАДКА ПРОГРАММЫ НА ЯЗЫКЕ АССЕМБЛЕРА

В состав программных средств, поставляемых как часть DOS, входит программа DEBUG (отладчик), с помощью которой может быть отлажена программа пользователя, написанная на языке Ассемблера. Обнаруженные при отладке ошибки могут быть легко исправлены либо непосредственной заменой содержимого ячеек памяти, либо заменой одной команды на другую с одновременной трансляцией ее в машинный код.

1. Запуск отладчика

Для запуска программы DEBUG с клавиатуры вводится следующая строка:
C> DEBUG [имя дисковода:] Имя файла.спецификация файла

Указывать имя файла при вызове программы DEBUG необязательно. В этом случае файл с программой пользователя может быть загружен после запуска с помощью команд N и L программы DEBUG.

После загрузки программы DEBUG без указания имени файла все сегментные регистры устанавливаются в состояния, соответствующие начальным адресам свободной памяти, следующей за областью памяти, занятой программой DEBUG. Счетчик команд IP получает значение 0100H. Регистры общего назначения и разряды регистра признаков устанавливаются в нуль. Указатель стека SP получает наименьшее значение из двух адресов: адреса конца сегмента стека загружаемой программы и адреса конца стека программы COMMAND.COM.

Если с помощью программы DEBUG загружается файл с расширением .EXE , то сегментный регистр CS , указатель стека SP и счетчик команд IP получают значения, определенные в загружаемом файле. Регистр CX будет содержать значение длины программы, подлежащей отладке, а если ее длина будет больше 64 Кбайт, то старшая часть значения длины будет записана в регистре BX .

В ответ на приглашение программы в виде черты, следует ввести одну из команд отладчика, полный перечень которых приведен в табл. 1.

	Назначение команды
A	Прямая трансляция в машинный код команды
C	Сравнение областей памяти одинакового размера
D	Вывод на экран дисплея содержимого памяти
E	Изменение содержимого памяти
F	Заполнение области памяти
G	Выполнение программы
H	Сложение и вычитание в шестнадцатеричном коде
I	Чтение и вывод на экран содержимого порта ввода
L	Загрузка с диска
M	Пересылка содержимого одной области памяти в
N	другую
O	Указание имени файла
Q	Запись данных в порт вывода
R	Выход в DOS
S	Изменение содержимого регистра и вывод на экран
T	Найти определенную строку
U	Команда трассировки (выполнение одной команды
W	программы)
	Команда дизассемблирования
	Запись на диск

2. Работа с отладчиком

Если при вызове отладчика файл с программой, подлежащей отладке, не указан, то загрузить его можно с помощью двух команд N и L. Сначала в команде N указываются место нахождения файла, его имя
и спецификация. Например,

```
- N E:SORT.EXE
```

Затем вводится команда L без указания параметров.

Обычно отладка начинается с просмотра программы. Для этого используется команда U , которая производит дизассемблирование, т.е. переводит машинный код в мнемонику языка Ассемблера. На экране дисплея после выполнения команды U отображаются адреса ячеек памяти, их содержимое (в шестнадцатеричном коде) и соответствующие мнемонические обозначения команд. При вызове команды U первый раз в ней следует указать адрес, с которого предполагается начать просмотр. Значение адреса определяется относительно сегментного регистра CS . На экране отображается восемь строк. При продолжении просмотра параметры в команде U можно не указывать. Пример выполнения команды U:

```
- U 0
 4A7F:0000 B86B4A          MOV AX, 4A6B
 4A7F:0003 8ED8            MOV DS, AX
```

Выполнение команды языка Ассемблера может быть осуществлено с помощью команды отладчика T , имеющей формат

T = АДРЕС

После выполнения каждой команды отлаживаемой программы на экран выводится содержимое всех регистров и значения разрядов регистра признаков. Например,

```
- T = 0
AX=4A6B BX=0000 CX=0040 DX=0000 SP=0BA8 BP=0100 SI=0000 DI=0000

DS=4AF2 ES=4AF2 SS=4AF2 CS=4A7F IP=0003 NV UP EI PL NA PO NC
4A7F:0003    MOV DS,AX
```

В следующих вызовах команды T параметр можно опустить. В этом случае выполняется следующая по порядку команда.

Программа пользователя может быть выполнена полностью или по частям с указанием адреса останова с помощью команды G , имеющей формат

```
G = АДРЕС АДРЕС
```

Выполнение отлаживаемой программы начинается с первого указанного в команде адреса, а останов осуществляется по второму.

С помощью команды R легко изменяется содержимое регистров. Для этого вводится команда R с указанием имени регистра. На экран выводится его содержимое и двоеточие, после которого может быть введено новое значение содержимого регистра. Например,

```
- R AX
AX 0000
:
```

Команда RF отображает значения всех разрядов регистра признаков и позволяет изменить значения любых из них. Мнемоническое обозначение признаков приведено в табл. 2.

Таблица 2 Содержимое регистра признаков

Значение признака	Обозначение 1 0
Переполнение (есть/нет)	OV NV
Направление (инкремент/декремент)	DN UP
Прерывание (разрешение/запрещение)	EI DI
Знак (минус/плюс)	NG PL
Результат (нуль/не нуль)	ZR NZ
Дополнительный перенос (да/нет)	AC NA
Паритет (четный/нечетный)	PE PO
Перенос (да/нет)	CY NC

Если в команде R имя регистра не указано, то на экран выводится содержимое всех регистров и признаков, а также адрес, код и мнемоническое обозначение очередной команды отлаживаемой программы.

Просмотр области памяти производится с помощью команды D , имеющей следующий формат:

D АДРЕС

При этом АДРЕС - значение начального адреса просматриваемой области памяти (64 байта). Для указания адреса следует сначала записать имя или значение сегментного регистра и после двоеточия текущий адрес в заданном сегменте. Например,

- D CS : 0

или

- D 4A7F : 0

Если в команде D сегмент не указан, то подразумевается адрес ячейки памяти, определенной относительно содержимого сегментного регистра DS .

Изменение содержимого ячейки памяти осуществляется с помощью команды E . Вызов этой команды производится аналогично вызову команды D. После вызова выполняется одно из следующих действий:

- нажимается клавиша ENTER, чтобы завершить команду E, оставив содержимое ячейки памяти неизменным;
- нажимается клавиша "пробел" для просмотра следующей ячейки памяти, оставив предыдущее значение неизменным;
- нажимается клавиша "минус", чтобы вернуться к предыдущей ячейке;
- вводится одна или две шестнадцатеричные цифры, которые определят новое содержимое ячейки памяти.

Описание остальных команд отладчика можно найти в специальных руководствах по MS DOS .

3. Порядок выполнения работы

1. С помощью отладчика проведите дизассемблирование Программы 1 и Программы 2 из Лабораторной работы № 1.

2. Сравните результат п.1 с листингом ассемблирования из Лабораторной раб № 1.

3. С помощью команды T проведите пошаговое выполнение программ 1 и 2, исправьте найденные ошибки.

4. Выполните сложение, вычитание, умножение и деление двух различных шестнадцатеричных чисел, выбранных индивидуально. Результат проверьте путем перевода чисел и результата действий в десятеричную систему исчисления.

ЛАБОРАТОРНАЯ РАБОТА № 3

ПРИМЕНЕНИЕ НЕКОТОРЫХ ФУНКЦИЙ ОПЕРАЦИОННОЙ СИСТЕМЫ

При разработке прикладной программы на языке Ассемблера следует иметь ввиду, что программа может быть эффективна, если известно, как связать ее выполнение с работой внешних устройств. В настоящей лабораторной работе необходимость такой связи рассматривается на примере выполнения программы сортировки. С этой целью выделим три части программы: запись в память конкретных значений элементов массива, подлежащих сортировке; собственно сортировку; завершение программы.

2. Внутренние прерывания

В персональных ЭВМ типа IBM PC, работающих с MS DOS, имеется набор служебных программ, позволяющих пользователю решать задачи управления файловой системой и устройствами ввода-вывода, обработки специальных ситуаций (например, завершение программы) и т.д. Обращение к служебным программам осуществляется с помощью внутренних прерываний MS DOS.

Для первоначального ознакомления с возможностями использования служебных программ ограничимся описанием некоторых функций прерывания, имеющего номер 21H.

Рассмотрим способы организации завершения программы. После того, как получен загрузочный модуль, остается поместить его в память ЭВМ и выполнить. В памяти конец программы ничем не отмечен и поэтому после выполнения последней команды загрузочного модуля будет выбираться следующая команда, код которой имеет произвольное значение (например, остался от предыдущей выполнявшейся программы). Разумеется, что дальнейшие действия программы будут непредсказуемыми. Не исключено, что придется осуществить перезагрузку операционной системы. Следовательно, необходимо использовать такие средства, которые позволили бы закончить выполнение прикладной программы и затем вернуть управление операционной системе.

С помощью прерывания 21H может быть выбрана одна из функций служебных программ. Вызов функции осуществляется в следующем порядке: сначала в регистр AH загружается номер функции, а затем с помощью команды Ассемблера INT 21H выполняется прерывание.

Функция с номером 4CH предназначена для организации завершения текущей программы и возвращения управления к DOS. Таким образом, в конце программы следует вставить следующие команды:

```
MOV AH, 4CH
INT 21H
```

С помощью функции 01H прерывания 21H может быть организован ввод данных с клавиатуры ЭВМ. При выполнении функции 01H осуществляется ожидание нажатия одной из клавиш клавиатуры. После нажатия клавиши соответствующий символ вводится в регистр AL и одновременно выводится на экран дисплея (ввод с эхом). (Подумайте и ответьте на вопрос: какой код будет введен в регистр AL, если нажата, например, клавиша с цифрой 9 ?).

Если вывод значения нажатой клавиши на экран не требуется, то применяется функция 08H (ввод с клавиатуры без эха).

Например, для заполнения массива данными вводом с клавиатуры может использоваться следующий фрагмент:

```
        MOV BX, OFFSET MAC
        PUSH BX
        MOV CX, NB
MM:     MOV AH, 01
        INT 21H
        MOV [BX], AL
        INC BX
        LOOP  MM
        POP  BX
```

С помощью прерывания 21H пользователю предоставляется возможность взаимодействия с клавиатурой, дисплеем, принтером, диском и асинхронным последовательным устройством. Некоторые наиболее часто используемые функции, вызываемые с помощью прерывания 21H, представлены ниже. Для вызова функции необходимо записать номер функции в регистр AH и выполнить прерывание INT 21H.

Функции для работы с клавиатурой

01H	Ввод символа с клавиатуры и изображение его на экране. Код символа после вызова функции загружается в регистр AL. Код проверяется на соответствие Ctrl-Break и при совпадении выполняется прерывание 23H. Для получения расширенного кода требуется выполнить два вызова этой функции.
06H	То же, что функция 01H, но без проверки на соответствие Ctrl- Break. Предварительно в DL необходимо записать значение FFH.
07H	То же, что функция 01H, но без изображения символа на экране и без проверки на соответствие Ctrl-Break.
08H	То же, что функция 07H, но с проверкой на соответствие Ctrl- Break.
0AH	Ввод строки символов с клавиатуры в буфер. Адрес буфера должен быть предварительно записан в DS:DX.
0CH	Очистка буфера клавиатуры. Затем выполняется функция, номер которой хранится в регистре AL (01H, 06H, 07H или 08H).

Функции для работы с дисплеем

02H Вывод символа на экран. Содержимое DL выводится на экран.
05H Вывод символа на печать. Содержимое регистра DL выводится на стандартное устройство печати.
09H Изображение строки на экране. Адрес начала строки должен быть записан в DS:DX. Текст строки должен заканчиваться символом $.

Функции для обмена данными с асинхронным последовательным устройством

03H Ожидание ввода символа через асинхронное последовательное устройство ввода. Код символа после ввода загружается в регистр AL.
04H Вывод символа через асинхронное последовательное устройство. Символ выводится из регистра DL.

Функции управления записью файлов

0DH Сброс текущего дисковода в начальное состояние.
0EH Задание нового текущего дисковода. В DL должен содержаться номер дисковода (0=A, 1=B, 2=C).
2EH Задание режима проверки. При вызове регистр DL должен содержать нуль, а регистр AL - нуль для отключения проверки, либо единицу для включения проверки. Эта функция используется тогда, когда требуется проверять правильность записи наиболее важных данных.

Функции для работы с датами и временем

2AH Чтение даты. CX - год, DH - месяц, DL - день.
2BH Установка даты. Для установки даты используются те же регистры, что и для функции 2AH.
2CH Чтение времени. CH - часы, CL - минуты, DH - секунды, DL - сотые доли секунды.
2DH Установка времени. Для установки времени используются те же регистры, что и для функции 2CH.

Функции для работы с векторами прерываний

25H Установка вектора прерывания. В DS:DX должно содержаться новое значение вектора прерывания.
35H Чтение вектора прерывания. В регистр AL заносится тип прерывания, а значение вектора читается в ES:BX.

2. Порядок выполнения работы

1. Напишите на языке Ассемблера программу сортировки методом "пузырька" и оформите ее в виде модуля с именем SORT. Стековый сегмент и сегмент данных объедините в группу (см. Приложение 3).

2. Получите .EXE-файл программы сортировки и загрузите его с помощью отладчика.

3. С помощью команды E отладчика заполните массив данными, подлежащими сортировке.

4. В пошаговом режиме отладьте программу, убедитесь в ее правильной работе.

5. Выйдите из отладчика в Norton и запустите программу сортировки. Наблюдайте за поведением компьютера и, если он "завис", осуществите перезагрузку операционной системы.

6. Дополните программу сортировки фрагментами ввода с клавиатуры и завершения программы. Прокомментируйте эти фрагменты. Убедитесь, что в этом случае программа завершается корректно.

ЛАБОРАТОРНАЯ РАБОТА № 4

ОРГАНИЗАЦИЯ РАБОТЫ С ВНЕШНИМИ УСТРОЙСТВАМИ. КОМАНДЫ ВВОДА-ВЫВОДА

В качестве модели внешнего устройства в настоящей работе используется звуковой генератор компьютера, состоящий из интервального таймера 8253 (аналог-КР580ВИ53), динамика и параллельного порта 8255 (аналог-КР580ВВ55), управляющего включением и выключением динамика.

1. Алгоритм управления

В компьютере типа IBM PC имеется таймер, второй счетчик которого настраивается на режим деления одной из частот генератора ЭВМ. Коэффициент деления выберем равным 1000. Тогда при значении тактовой частоты 1.19 МГц на выходе счетчика 2 получается звуковая частота 1190 Гц. Для настройки счетчика 2 на режим деления в регистр управляющего слова таймера следует записать значение B6H по адресу 43H. Загрузка коэффициента деления 1000 в счетчик 2 по адресу 42H производится в два приема. Сначала загружается младший байт шестнадцатеричного эквивалента числа 1000, а затем - старший байт числа. Пересылка данных в счетчик 2 по адресу 42H может быть осуществлена с помощью команды OUT 42H,AL . Первый раз пересылается значение E8H (младший байт), а перед тем, как применить команду во второй раз, в регистр AL необходимо переслать старший байт числа.

Сигнал звуковой частоты с выхода счетчика 2 через элемент, выполняющий функцию ключа, поступает на динамик. Управление ключом осуществляется с помощью сигналов, поступающих с нулевого и первого разрядов параллельного порта, имеющего адрес 61H. Другие шесть разрядов порта устанавливаются

операционной системой и их без нарушения функционирования ЭВМ изменять нельзя.

Перед тем как запрограммировать счетчик таймера в нужный режим, необходимо на всякий случай выключить динамик. Для этого в два младших разряда порта засылаются нули. Рассмотрим последовательность действий, предназначенных для выключения динамика. Сначала, чтобы не нарушить работу компьютера, с помощью команды IN производится опрос состояния порта и одновременное запоминания его значения, например, в регистре AL. Затем для установки двух младших разрядов регистра AL используется логическая команда AND, выполняющая операцию логического поразрядного умножения. Удобно применить эту команду с непосрдственной адресацией, когда в качестве операнда используется маска (чему она равна ?). После пересылки в порт 61H результата выполнения команды AND произойдет выключение динамика без нарушения функционирования всей ЭВМ.

Включение динамика производится после программирования счетчика 2 установкой двух младших разрядов порта 61H в единицы. Способ манипуляции с двумя младшими разрядами байта остается тем же, что и при выключении динамика. Но вместо логической команды AND используется команда OR (логическое поразрядное сложение) с непосредственным операндом, в двух младших разрядах которого записаны единицы.

Длительность звучания определяется временем задержки и реализуется программно. Для этого, например, в регистр CX заносится переменная, задающая длительность звучания. Если занести в регистр CX нуль и воспользоваться командой LOOP, то при первом же ее выполнении осуществится декремент регистра CX, и его содержимое станет FFFFH. Затем декремент содержимого регистра CX будет продолжаться до тех пор, пока регистр не обнулится, и цикл, организованный с помощью команды LOOP, не прекратится. Этот фрагмент может выглядеть следующим образом:

```
        SUB CX,CX
DELAY:  LOOP DELAY
```

После этого динамик надо выключить.

2. Порядок выполнения работы

1. Напишите программу управления динамиком по алгоритму из п.1. Оформите ее в виде процедуры, размещенной в программном модуле, например, с именем MOD1, и доступной из других модулей.
2. Модифицируйте модуль сортировки так, что завершение выстраивания элементов массива в порядке возрастания, отмечается звуковым сигналом.
3. Транслируйте оба модуля отдельно, а затем свяжите их в одну программу с помощью редактора связей.

ЛАБОРАТОРНАЯ РАБОТА № 5

ПРИМЕНЕНИЕ СРЕДСТВ БАЗОВОЙ СИСТЕМЫ ВВОДА-ВЫВОДА. ЧАСТЬ 1

1. Назначение базовой системы ввода-вывода

Базовая система ввода-вывода (БСВВ) (английское сокращение BIOS) содержит набор программ для обслуживания основных внешних устройств. Применение средств BIOS в прикладных программах в некоторых случаях позволяет добиться большей эффективности по сравнению с использованием функций операционной системы. Программы BIOS размещаются в блоке ПЗУ емкостью 8 Кбайт. Адрес блока ПЗУ определяется относительно сегментного адреса FE00H.

Для вызова программы BIOS используются векторы прерываний, имеющие номера от 01H до 1FH. Приведем примеры некоторых функций:

10H функции экрана;
11H процедура определения присоединенного оборудования;
12H процедура определения объема ОЗУ;
16H функции клавиатуры;
17H функции печати;
19H функции начальной загрузки.

2. Функции экрана

BIOS предоставляет пользователю ряд функций для работы с экраном. Эти функции вызываются с помощью внутреннего прерывания INT 10H. Номер функции загружается в регистр AH. В таблице содержится список функций с указанием используемых регистров до вызова функции и после ее выполнения.

Таблица

Функции BIOS для работы с экраном

Номер	Назначение	До вызова	После вызова
1	2	3	4
00H	Установить режим экрана	AL - номер режима	
01H	Установить размер курсора	CH - верхняя линия курсора CL - нижняя линия курсора	
02H	Установить позицию курсора	BH - номер страницы памяти дисплея DH - строка DL - столбец	

03H	Сведения о позиции и размере курсора	BH - номер страницы памяти дисплея	H- верхняя линия курсора L – нижняя линия курсора H- номер строки L- номер столбеца
05H	Установить активную страницу	AL - номер страницы памяти дисплея	
06H	Переместить окно вверх	AL-число строк сдвига (при AL=00 окно стирается) BH- атрибут символа CH- номер верхней строки окна CL-номер первого столбца окна DH-номер нижней	
07H	Переместить окно вниз	DL-номер правого столбца окна	
08H	Сведения о символе и атрибуте	BH - номер страницы памяти дисплея	H- символ L- атрибут
09H	Записать символ и атрибут	AL - символ BL – атрибута BH - номер страницы CX - повторитель	
0AH	Записать символ в текущую позицию курсора		
0BH	Установить палитру	BH - номер палитры BL - основной цвет	
0CH	Записать точку	AL - цвет CX - колонка DL - линия	
0DH	Сведения о точке	CX - колонка DL - линия	AL - цвет точки
0FH	Сведения о режиме экрана		AH - ширина экрана AL - номер режима BH - номер страницы

Иллюстрацию возможностей работы с экраном дисплея рассмотрим на примере вывода результатов, полученных в процессе выполнения программы сортировки. Отображение данных будем осуществлять на фоне выделенного цветом прямоугольника (окна)

Формирование окна в заданном месте экрана может быть реализовано с помощью вызова функции 07H прерывания 10H BIOS. Но прежде, чем это сделать, необходимо выбрать режим работы экрана. В зависимости от типа дисплея может быть выбран текстовый или графический, черно-белый или цветной режим, а также может быть выбрано число столбцов и строк экрана. В данной работе воспользуемся тем режимом, который устанавливается при загрузке операционной системы. Для IBM PC устанавливается текстовый цветной режим с отображением знаков в поле, состоящем из 80 столбцов и 25 строк (от 0 до 79 и от 0 до 24 соответственно).

Перед тем, как сформировать окно, необходимо очистить экран от всех знаков, оставшихся от выполнения предыдущих программ. Это достигается с помощью сдвига всего окна экрана за его пределы. С этой целью вызывается функция 07H прерывания 10H, а в регистры AL и CX предварительно заносятся нули (зачем ?). Одновременно устанавливается цвет фона очищенного экрана. Для установки цвета используется следующее кодирование:

000 - черный	100 - красный
001 - синий	101 - оранжевый
010 - зеленый	110 - коричневый
011 - фиолетовый	111 - белый.

Приведенные коды записываются в следующие разряды байта, предназначенного для указания атрибута символа: разряды 0, 1, 2 - цвет символа; разряды 4, 5, 6 - цвет фона поля символа.

Третий разряд байта атрибута устанавливается в нуль для указания низкой интенсивности отображения символа и в единицу - для высокой. Седьмой разряд устанавливается в нуль для указания нормального изображения символа и в единицу - для мерцающего. Значение атрибута символа загружается в регистр BH. При формировании окна его заполнение производится знаком пробела. Если при очистке экрана предварительно в регистр BH записать число 10H, то будет установлен фон синего цвета.

Для формирования окна, в котором будут отображаться данные, необходимо задать координаты. Чтобы окно находилось в середине экрана и по его высоте умещалось бы, например, девять символов, следует выбрать значения координат верхнего угла 5,28, а нижнего - 15,50. Первая цифра задает номер строки, а вторая - номер столбца. Цвет фона окна можно выбрать зеленым.

Фрагмент программы, которая производит очистку экрана и формирование окна, приведен ниже:

```
MOV   AX, 0700H
MOV   BH, 10H
MOV   CX, 0
MOV   DH, 24
MOV   DL, 79
INT   10H
```

```
MOV  AX, 0700H
MOV  BH, 20H
MOV  CH, 05
MOV  CL, 28
MOV  DH, 15
MOV  DL, 50
INT  10H
```

3. Порядок выполнения работы

1. Прокомментируйте приведенную в п.2 программу. Оформите ее в виде процедуры, входящей в состав отдельного модуля и доступной из главного модуля программы сортировки.
2. Модифицируйте модуль сортировки так, чтобы перед началом ввода данных производилась очистка экрана и формирование окна.
3. Проведите трансляцию и компоновку программы. Проверьте ее работу.

ЛАБОРАТОРНАЯ РАБОТА № 6

ПРИМЕНЕНИЕ СРЕДСТВ БАЗОВОЙ СИСТЕМЫ ВВОДА-ВЫВОДА. ЧАСТЬ 2

Отображение символов, введенных с клавиатуры дисплея и подлежащих сортировке, а также упорядоченных по возрастанию соответствующих кодов символы, будем осуществлять в заданных строках и столбцах экрана. При этом неупорядоченные и упорядоченные элементы расположим в соответствующих вертикалях, отделенных друг от друга несколькими пробелами. Отображаемые элементы укажем в виде символов на фоне, отличающемся от фона окна, например, красные буквы на белом фоне (цвет фона окна - зеленый).

1. Заполнение буфера экрана

Для отображения символов в поле сформированного окна воспользуемся непосредственной загрузкой ячеек памяти буфера экрана. На экран дисплея выводится 25 строк текста по 80 символов в каждой строке, то есть всего 2000 символов. Позиции символов на экране нумеруются от 0 до 1999, причем каждому символу соответствует своя ячейка памяти в буфере экрана. Для цветного режима на отображение дного символа требуется два байта, в первом из которых записывается код отображаемого символа, а во втором - код атрибута символа. В атрибуте символа указывают цвет, фон, интенсивность и мерцание.

Вычисление (в десятичной системе) адреса ячейки памяти для записи отображаемого знака производится по формуле

адрес = 4096*P + 160*L + 2*C,

где:

P - номер страницы буфера экрана;

L - номер строки экрана;

C - номер столбца экрана.

Для точного определения физического адреса ячейки памяти буфера следует организовать сегмент данных (например, в экстра сегменте) с сегментным адресом B800H, то есть к вычисленному адресу ячейки памяти буфера экрана добавляется значение B8000H.

Пусть, например, первый символ располагается на пересечении 6-ой строки и 32-го столбца. Тогда для отображения первого символа на нулевой странице экрана адрес ячейки памяти буфера экрана должен иметь значение 160*6 + 32*2 = 1024, что соответствует числу 0400H, которое в главном модуле программы сортировки загружается в индексный регистр DI. Теперь, если переслать код введенного с клавиатуры символа или содержимое ячейки памяти, где хранится значение элемента упорядоченного массива, в ячейку памяти с адресом B8400H, а в ячейку памяти с адресом B8401H атрибут, то на экране в поле окна отобразится соответствующий символ.

Для уменьшения помех на экране заполнение буфера экрана производится в момент вертикального обратного хода луча электронно-лучевой трубки. Этот момент определяется с помощью анализа состояния регистра адаптера дисплея, имеющего адрес порта 03DAH. Если нулевой разряд регистра состояния установлен в единицу, то возможен доступ к буферу экрана.

Фрагмент программы, которая реализует описанные выше действия, представлен ниже:

```
        MOV  DX, 03DAH
WW:  IN      AL, DX
        TEST  AL, 01
        JNZ    WW
        MOV  AL, [BX]
        MOV  ES:[DI], AL
        MOV  AL, COLOR
        MOV  ES:[DI+1], AL
        ADD   DI, 160
```

В приведенном фрагменте переменной COLOR соответствует атрибут символа, передаваемый из основной программы. По адресу [BX] записан код отображаемого символа. В сегментный регистр ES предварительно (в главном модуле) должен быть загружен адрес B800H. В конце программы содержимое индексного регистра DI увеличивается на 160, чтобы при повторном вызове процедуры символ отображался на следующей строке.

2. Порядок выполнения работы

1. Оформите программу заполнения буфера экрана в виде процедуры, входящей в состав самостоятельного модуля и доступной из главного модуля программы сортировки. Прокомментируйте программу.

2. В главном модуле программы сортировки вызов процедуры осуществите из двух позиций. В первом случае - при вводе данных с клавиатуры. Тогда символы нажатых клавиш будут отображаться в левом столбце окна (состояние до начала сортировки). В правом столбце окна отображаются результаты упорядочивания. Для этого процедура вызывается в конце главного модуля. Для того, чтобы столбцы были отделены пробелами, содержимое регистра DI перед вторым вызовом процедуры требуется увеличить. Кроме того, обратите внимание, что процедуру следует вызывать в цикле, а перед циклом в обоих случаях необходимо подготовить регистры ES и DI. Опишите переменную COLOR и задайте ее значение.

3. Модифицируйте главный модуль так, чтобы перед возвратом в среду системы Norton осуществлялась временная задержка для просмотра результатов.

4. Проведите трансляцию модулей и с помощью редактора связей скомпонуйте главный модуль и модули, осуществляющие ввод и вывод информации. С помощью отладчика DEBUG добейтесь правильной работы программы.

Л И Т Е Р А Т У Р А

1. Юров В. Assembler/ - СПб: «Питер», 2000.- 624 с.

2. Финогенов К.Г. Основы языка Ассемблера. – М.: Радио и связь, 1999. – 288 с.

3. Абель Питер Язык Ассемблера для IBM PC и программирования. – М.: Высш.шк., 1992.-447 с.

4. Нортон П., Соухе Д. Язык ассемблера для IBM PC.-М.: Компютер, 1993.- 352 с.

5. Джорден Р. Справочник программиста PC типа IBM PC, XT и AT. М.: Фин. и статистика, 1991. - 544 с.

Приложение 1

СПИСОК ДИРЕКТИВ, ОПЕРАТОРОВ И АТРИБУТОВ АССЕМБЛЕРА

Директивы

ASSUME	Связывает имя сегмента с сегментным регистром.
DB	Определяет и резервирует требуемое число байтов.
DW	Определяет и резервирует требуемое число слов.
DD	Определяет и резервирует требуемое число двойных слов.
DQ	Определяет и резервирует требуемое число слов четырехкратной длины.
DT	Определяет и резервирует требуемое число слов, имеющих длину десять байт.
END	Определяет конец текста программы.
ENDP	Определяет конец текста процедуры.
ENDS	Определяет конец сегмента.
EQU	Ставит в соответствие символическому имени значение выражения.
=	Ставит в соответствие символическому имени значение в разных местах программы
EXTRN	Описывает символические имена, определенные в другом модуле.
GROUP	Объединяет сегменты, расположенные в различных модулях.
INCLUDE	При трансляции включает программу из другого файла в исходную программу.
LABEL	Приписывает символическому имени указанные атрибуты.
NAME	Задает имя модуля
ORG	Переопределяет текущее значение счетчика (PC) IP.
PROC	Определяет имя процедуры, вызываемой оператором CALL.
PUBLIC	Определяет символические имена, на которые можно ссылаться в других модулях
RADIX	Отменяет необходимость использования указателей B и H в числах.
RECORD	Организует битовые поля внутри байта или слова
SEGMENT	Объявляет сегментом совокупность ассемблерных строк
STRUC	Организует поля внутри строки, составленной из байтов.

Операторы

PTR	Устанавливает тип данных BYTE, WORD, DWORD или тип перехода FAR или NEAR.
LENGTH	Длина (число байтов, слов или двойных слов).
SIZE	Размер (общее число байт).
TYPE	Тип переменной.
OFFSET	Вычисляет смещение символического имени относительно начала сегмента.
RS:	Оператор замены сегмента (RS -имя сегментного регистра).
SHORT	Формирование короткого смещения при переходах (-128 - +127 б.).

Атрибуты объединения

PUBLIC	Длина объединенного сегмента равна сумме длин объединяемых сегментов.
COMMON	Длина объединенного сегмента равна наибольшей из длин объединяемых сегментов.
STACK	Длина сегмента стека равна сумме объединяемых сегментов.
MEMORY	Вызывает размещение сегмента данных после сегмента кода (размер сегмента определяется аналогично объединению COMMON).

Атрибуты типа выравнивания

PARA	Адрес границы сегмента кратен 16.
BYTE	Граница сегмента может начинаться с любого адреса.
WORD	Граница сегмента начинается с четного адреса.
PAGE	Адрес границы сегмента кратен 256 (страница).
INPAGE	Сегмент размещается внутри страницы (не более 256 байтов).

Приложение 2

СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА 8086

Условные обозначения

R	регистр общего назначения: AX (AH,AL); BX (BH,BL); CX (CH,CL); DX (DH,DL); SI,DI; SP,BP.
RS	сегментные регистры CS,DS,SS,ES.
ac	аккумулятор (AL или AX).
ata	непосредственные данные (8- или 16-разрядный операнд).
M	адрес ячейки памяти.
M	адрес ячейки памяти.

Команды

AAA;AAD;AAM;AAS - коррекция результата сложения, деления, умножения и вычитания данных, соответственно, представленных в коде ASCII;

ADC R,R; ADC R,M;
ADC R,DATA; ADC ac,data;
ADC M,R; ADC M,data - сложение с переносом;

ADD R,R; ADD R,M;
ADD R,data; ADD ac,data;
ADD M,R; ADD M,data - сложение;

AND R,R; AND R,M;
AND R,data;AND ac,data;
AND M,R; AND M, data - поразрядное логическое умножение;

CALL NEAR PTR МЕТКА - вызов процедуры внутрисегментный прямой;
CALL FAR PTR МЕТКА - то же межсегментный прямой;
CALL WORD PTR ОПЕРАНД - то же внутрисегментный косвенный;
CALL DWORD PTR ОПЕРАНД - то же межсегментный косвенный;

CBW - преобразование байта в слово;
CLC - сброс признака переноса;
CLD - сброс признака направления;
CMC - инвертирование признака переноса;
CLI - сброс признака прерывания;

CMP R,R; CMP M,R;
CMP R,data; CMP ac,data;
CMP R,M; CMP M,data - сравнение двух операндов;

CMPSB, CMPSW - сравнение строк байтов или слов соответственно;
CWD - преобразование слова в двойное слово;
DAA, DAS - коррекция результатов сложения и вычитания данных, соответственно, представленных в упакованном двоичнодесятичном коде;
DEC R; DEC M - декремент;
DIV R; DIV M - деление чисел без знака;
ESC - выборка кода и операнда для сопроцессора 8087;
HLT - останов;
IDIV R; IDIV M - деление чисел со знаком;
IMUL R; IMUL M - умножение чисел со знаком;
IN ac,data; IN ac,DX - ввод байта или слова из порта;
INC R; INC M - инкремент;
INTO - прерывание по переполнению;
IRET - возврат после обработки прерывания;

Перейти, если:

JA/JNBE - выше/не ниже или равно;
JAE/JNB - выше или равно/не ниже;
JB/JNAE - ниже/не выше или равно;
JBE/JNA - ниже или равно/не выше;
JC - есть перенос;
JE/JZ - равно/нуль;
JG/JNLE - больше/не меньше или равно;
JGE/JNL - больше или равно/не меньше;
JL/JNGE - меньше/не больше или равно;
JLE/JNG - меньше или равно/не больше;
JNC - нет переноса;
JNE/JNZ - не равно/не нуль;
JNO - нет переполнения;
JNP/JPO - нет паритета/паритет нечетный;
JNS - нет знака;
JO - есть переполнение;
JP/JPE - есть паритет/паритет четный;
JS - есть знак;
JCXZ - содержимое регистра CX=0;

(Примечание: 1) термины "больше" и "меньше" относятся к знаковым числам, представленным в дополнительном коде, а "выше" и "ниже" - к беззнаковым; 2) все команды условных переходов передают управление по адресу в диапазоне от -128 до +127 байт; для реализации переходов по адресам, превышающим указанный диапазон, применяют комбинацию команд условного и безусловного переходов).

Безусловные переходы:

JMP SHORT LAB - внутрисегментный (в/с) к метке LAB;
JMP LAB - в/с к метке LAB в пределах от -128 до +127 байт;
JMP NEAR PTR LAB - в/с к метке LAB в любую точку сегмента;
JMP WORD PTR VAR - то же, но адрес перехода записан в ячейке VAR;
JMP WORD PTR BX - то же, но адрес перехода записан в регистре BX;
JMP WORD PTR [BX] - в/с в любую точку сегмента по адресу, хранящемуся в ячейке, адрес которой записан в регистре BX;
JMP FAR PTR LAB - межсегментный (м/с) к метке LAB в любую точку другого сегмента;
JMP DWORD PTR [BX] - м/с косвенный;
JMP DWORD PTR VAR - м/с прямой (адрес перехода записан в ячейке VAR);
LAHF - загрузка регистра AH содержимым регистра признаков;
LDS - загрузка указателя с использованием DS;
LEA - загрузка эффективного (исполнительного) адреса;
LES - загрузка указателя с использованием ES;
LOCK - захват шины;
LODSB,LODSW - загрузка строки байтов или слов;
LOOP LAB - цикл по состоянию регистра CX до метки LAB;
LOOPE/LOOPZ - зациклить, если (CX)=0/ZF=1;
LOOPNE/LOOPNZ - зациклить, если (CX)<>0/ZF=0;
MOV R,R; MOV R,RS; MOV R,M;
MOV R,data; MOV RS,R; MOV M,R;
MOV M,data - команды пересылки;
MOVS - переслать цепочку;
MOVSB, MOVSW - загрузить цепочку байт, слов;
MUL R, MUL M - умножение чисел без знака;
NEG R, NEG M - преобразование в дополнительный код;
NOP - пустая операция;
NOT R, NOT M - логическое отрицание;
OR R,R; OR R,data;
OR ac,data; OR R,M; OR M,data - логическое поразрядное ИЛИ;
OUT data,ac; OUT DX,ac - вывод байта или слова;
POP R; POP RS; POP M - чтение из стека;
POPF - чтение из стека флажков;
PUSH R; PUSH RS; PUSH M - запись в стек;
PUSHF - запись в стек флажков;
RCL R,1; RCL M,1 - циклический сдвиг влево через перенос на 1;
RCL R,CL; RCL M,CL - то же, но сдвиг произвольный;
RCR R,1; RCR M,1; RCR R,CL; RCR M,CL - то же вправо;
REP, REPE/REPZ, REPNE/REPNZ - повторение операций при обработке строк;
RET, RET data - возврат из процедуры;
ROL R,1; ROL R,CL; ROL M,1; ROL M,CL - циклический сдвиг влево;
ROR R,1; ROR R,CL; ROR M,1; ROR M,CL - то же вправо;

SAHF - запись содержимого AH в регистр признаков;
SHL R,1; SHL R,CL; SHL M,1; SHL M,CL - арифметический сдвиг влево;
SHR R,1; SHR R,CL; SHR M,1; SHR M,CL - то же вправо;
SBB R,R; SBB R,M; SBB M,R;
SBB R,data; SBB M,data - вычитание с заемом;
SCASB; SCASW - поиск байта или слова в строке;
SHL R,1; SHL M,1; SHL R,CL; SHL M,CL - логический сдвиг влево;
SHR R,1; SHR M,1; SHR R,CL; SHR M,CL - то же вправо;
STC; STD; STI - установка в 1 признаков переноса, направления и прерывания соответственно;
STOSB; STOSW - запоминание строки байтов или слов;
SUB R,R; SUB R,M; SUB R,data; SUB M,R; SUB M,data
- вычитание;
TEST R,R; TEST R,M; TEST R,data; TEST ac,data; TEST M,R; TEST M,data
- поразрядное И с сохранением значений операндов;
WAIT - ожидание;
XCHG R,R; XCHG R,M;
XCHG ac,R; XCHG M,R; XCHG R,ac - обмен значениями операндов;
XLAT - организация работы с таблицами;
XOR R,R; XOR R,M; XOR R,data;
XOR ac,data; XOR M,R; XOR M,data - поразрядное исключающее ИЛИ.

Приложение 3

Текст программы сортировки

Если Вы не хотите писать программу самостоятельно, то можете воспользоваться фрагментом, приведенным ниже. Однако этот фрагмент содержит две ошибки, которые следует обнаружить с помощью отладчика.

В программе: NB - количество элементов типа байт, адреса которых задаются через регистр BX.

```
SORT: MOV  DX, NB
      PUSH BX
      PUSH DX
      XOR  CL, CL
COMP:       MOV  AL, [BX]
      CMP  AL, [BX+1]
      JLE  NEXT
      XCHG AL, [BX+1]
      MOV  [BX], AL
      MOV  CL, 0FFH
NEXT:INC   BX
      DEC  DX
      JNZ  COMP
      POP  DX
      POP  BX
      OR   CL, CL
      JNZ  SORT
```

АППАРАТНЫЕ И ПРОГРАММНЫЕ СРЕДСТВА.
IBM-СОВМЕСТИМЫХ КОМПЬЮТЕРОВ

Методические указания к лабораторным работам
по курсу "ЭВМ в системах управления"
для студентов специальности 21.02

Составители: А.Л. Калабин, Г.А. Дмитриев, А.А. Шейнман
Редактор Т.М. Маренко
Технический редактор Г.В. Комарова

Подписано в печать 16.03.98

Формат 60x84 1/16		Бумага писчая
Физ.печ.л. 8,25	Усл.печ.л. 2,09	Уч.-изд.л. 1,95
Тираж 100 экз.	Заказ № 22	С - 545

Тверь. Типография ТГТУ