А. В. Карпов

СОЗДАНИЕ И ВЫПОЛНЕНИЕ ПРОСТЫХ ПРОГРАММ НА АССЕМБЛЕРЕ МП i8086 (часть II)

Методические указания к лабораторной работе

Волгоград 2006

УДК 681.325

Создание и выполнение простых программ на Ассемблере МП i8086 (часть II): Методические указания к лабораторной работе/ Сост. А. В. Карпов. -- Волгоград, 2006.-- 10 с.

Приведено описание лабораторной работы “Создание и выполнение простых программ на Ассемблере МП i8086” по курсу “Вычислительные машины, системы и сети”, в которой изучаются директивы языка Ассемблер МП i8086 и функции INT 21h DOS. Издание предназначено для студентов дневной, вечерней и заочной форм обучения специальности 2102.

Цель работы

Целью настоящей работы изучение директив языка, элементов и структуры простейших программ на Ассемблере МП i8086 и функций INT 21h DOS.

1 Директивы Ассемблера (продолжение)

Инициализированные данные

Инициализация массивов

В одной директиве определения данных может указываться несколько значений. Например, директива:

```
SampleArray          DW 0, 1, 2, 3, 4
```

создает массив из пяти элементов с именем ***SampleArray***, элементы которого имеют размер в слово. В директивах определения данных можно использовать любое число значений, умещающееся на строке.

Если необходимо определить массив, который слишком велик и не может уместиться на одной строке, то для этого просто нужно добавить несколько строк. Метку в директиве определения данных указывать необязательно. Например, по директивам:

```
SquareArray DD 0, 1, 4, 9, 16
            DD 25, 36, 49, 64, 81
            DD 100, 121, 144, 169, 196
```

создается массив элементов размером в двойное слово с именем ***SquareArray***, состоящий из 15 целых чисел.

Ассемблер позволяет определить блок памяти, инициализированный указанным значением, с помощью операции DUP. Например:

```
BlankArray DW 100h DUP (0)
```

Здесь создается массив ***BlankArray***, состоящий из 255 слов, инициализированных значением 0. Аналогично, директива:

```
ArrayOfA DB 92 DUP ('A')
```

создает массив из 92 байт, каждый из которых инициализирован символом A.

Неинициализированные данные

Иногда нет смысла присваивать переменной памяти начальное значение. Предположим, например, что программа считывает следующие 10 набранных на клавиатуре символов и записывает их в массив с именем KeyBuffer:

```
MOV CX, 10
MOV BX, OFFSET KeyBuffer
GetKeyLoop:
MOV AH, 1                    ; функция DOS ввода с клавиатуры
```

```
INT 21h                    ; получить следующий нажатый символ
MOV [BX],AL
INC CX
loop GetKeyLoop
```

Инициализировать KeyBuffer можно следующим образом:

```
KeyBuffer DB 10 DUP (0)
```

но смысла в этом немного, так как начальные значения массива ***KeyBuffer*** будут немедленно перезаписаны в цикле ***GetKeyLoop***. Все, что здесь в действительности нужно - это определить переменную в памяти, как неинициализированную. Ассемблер предусматривает такую возможность с помощью символа "знак вопроса" (?).

Вопросительный знак указывает Ассемблеру, что резервируется ячейка памяти, но она не инициализируется. Например, в последнем примере ***KeyBuffer*** можно определить так:

```
KeyBuffer DB 10 DUP (?)
```

В данной строке резервируется 10 байт памяти, начиная с метки ***Keybuffer***, но этим байтам не присваивается никакого конкретного значения.

Однако, каждый раз, перед использованием неинициализированной переменной в памяти в качестве операнда-источника, надо быть уверенным в том, что она была инициализирована в программе перед тем, как использоваться. Например, было бы ошибочным использовать содержимое ***KeyBuffer*** в последнем примере до того, как этот массив будет заполнен, так как находящиеся в ***KeyBuffer*** начальные значения не определены.

Именованные ячейки памяти

Другой удобный способ присваивать имя ячейке памяти предоставляет директива LABEL.

Директива LABEL позволяет определить как имя метки, так и ее тип, не определяя при этом данные. Например, еще один способ, с помощью которого можно определить в предыдущем примере массив ***KeyBuffer***:

```
KeyBuffer LABEL BYTE
DB 10 DUP (?)
```

Типы меток, которые можно определить с помощью директивы LABEL, включают в себя:

BYTE	PWORD	FAR
WORD	QWORD	PROC
DWORD	TBYTE	UNKNOWN
FWORD	NEAR.	

Типы BYTE (байт), WORD (слово), DWORD (двойное слово), PWORD, QWORD и TBYTE определяют, соответственно, 1-, 2-, 4-, 6-, 8- и 10-байтовые элементы данных. Пример инициализации переменной памяти, как пары байт, и обращения к ней, как к слову:

```
dseg SEGMENT ‘DATA’
WordVar LABEL WORD
DB 1,2
dseg ENDS
cseg SEGMENT ‘CODE’
...
MOV AX, [WordVar]
```

Когда эта программа выполняется, в регистр AL загружается 1 (первый байт ***WordVar***), а в регистр AH - значение 2.

2 Прерывание INT 21H (Функции DOS)

Прерывание 21H носит название "диспетчер функций". Диспетчер функций отвечает за выполнение большей части работы MS-DOS.

Функции DOS обычно используются (где это возможно) для таких операций, как ввод с клавиатуры или из файла, вывод на экран или в файл и печать информации.

Поскольку не все IBM PC-совместимые компьютеры одинаковы, DOS часто скрывает различия между ними. Если игнорировать в программах функции DOS и выходить непосредственно на аппаратуру, такие программы могут не работать на других компьютерах. Кроме того, программы, работающие в обход DOS, вероятнее всего не смогут сосуществовать с другими резидентными в памяти программами.

Чтобы программно обратиться к системной функции, необходимо выполнить следующее:

записать номер соответствующей функции в регистр AH;

записать в соответствующие регистры параметры, необходимые для работы функции (если это требуется);

вызвать прерывание 21H.

При обращении к прерыванию 21H, управление передается MS-DOS. Операционная система по значению регистра AH определяет, какая функция должна выполняться. Затем из остальных (вполне определенных для каждой функции) регистров считываются значения параметров, после чего требуемая функция выполняется. MS-DOS помещает возвращаемые функцией параметры в соответствующие регистры и возвращает управление в вызывающую программу.

Примечание. В том случае, когда DOS не обеспечивает нужную вам функцию, можно использовать функцию BIOS. Если же нет и ее, обращаться к аппаратуре непосредственно.

3 Порядок выполнения работы

1. Напишите программу №1, имеющую тип .EXE выводящую на экран сообщение

Доброе утро!

если текущее системное время до 12.00, и выводящую сообщение

Добрый день!

если текущее системное время после 12.00.

2. Отладьте программу №1.

3. Создайте исполняемый файл программы №1 типа .EXE.

4. Запустите программу №1 на выполнение.

5. Напишите программу №2 выполняющую те же действия что и программа №1, но имеющую тип .COM.

6. Отладьте программу №2.

7. Создайте исполняемый файл программы №2 типа .COM.

8. Запустите программу №2 на выполнение.

9. Напишите программу №3, имеющую тип .EXE выводящую на экран запрос

Сейчас время до полудня?

и при нажатии клавиш “Y” или “y” выводящую сообщение

Доброе утро!

при нажатии клавиш “N” или “n” выводящую сообщение

Добрый день!

при нажатии других клавиш выводящую запрос.

10. Отладьте программу №3.

11. Создайте исполняемый файл программы №3 типа .EXE.

12. Запустите программу №3 на выполнение.

13. Напишите программу №4 выполняющую те же действия что и программа №3, но имеющую тип .COM.

14. Отладьте программу №4.

15. Создайте исполняемый файл программы №4 типа .COM.

16. Запустите программу №4 на выполнение.

4 Содержание отчета

Отчет о лабораторной работе должен содержать следующие сведения: 1) цель работы; 2) текст программ.

5 Контрольные вопросы

1. Перечислите известные Вам директивы Ассемблера МП i8086.

2. Перечислите известные Вам функции прерывания INT 21h.

Приложение. ФУНКЦИИ ПРЕРЫВАНИЯ INT 21H (DOS)

Вызов функции

1. В регистр AH записать номер функции.
2. Другие регистры - в соответствии с конкретной функцией.
3. Вызвать прерывание INT 21h. Прерывание INT 21h заставляет систему выполнять функцию DOS, номер которой записан в регистре AH.

Пример: Вызов функции «Получить время»

```
MOV AH, 2Ch     ; 2Ch - номер функции «Получить время»
INT 21h         ; Вызов функции
```

Функции

ЧИТАТЬ С КЛАВИАТУРЫ И ЭХО (ФУНКЦИЯ 01H)

Вызов: AH=01h

Возвращает: AL - введенный символ

Функция 01h ожидает ввода символа со стандартного ввода, затем выдает эхо-символ на стандартный вывод и возвращает его в регистре AL.

ОТОБРАЗИТЬ СИМВОЛ (ФУНКЦИЯ 02H)

Вызов: AH=02h

DL - отображаемый символ

Возвращает: --

Функция 02h посылает символ из регистра DL на стандартный вывод.

ПРЯМОЙ ВВОД/ВЫВОД С КОНСОЛИ (ФУНКЦИЯ 06H)

Вызов: AH=06h

DL - См. ниже

Возвращает: AL

Если DL перед вызовом функции был равен FFh, то сброс флага переноса означает, что в AL содержится символ со стандартного ввода. Установка CF означает, что такого символа не было и AL=0.

Действие Функции 06H зависит от содержимого регистра DL:

Значение DL	Действие
FFh	Если символ был прочитан со стандартного ввода, то он возвращается в AL, а CF сбрасывается (0). Если символ не был прочитан, то CF устанавливается (1)
Любое, кроме FFh	Символ из регистра DL посылается на стандартный вывод

ПРЯМОЙ ВВОД С КОНСОЛИ (ФУНКЦИЯ 07H)

Вызов: AH=07h

Возвращает: AL - символ с клавиатуры

Функция 07h ждет ввода символа со стандартного ввода и возвращает его в AL. Эта функция не дает эхо символа.

ЧИТАТЬ С КЛАВИАТУРЫ (ФУНКЦИЯ 08H)

Вызов: AH=08h

Возврат: AL - символ с клавиатуры

Функция 08h ждет символ с клавиатуры и возвращает его в AL. Эта Функция не дает эхо символа.

ВВОД С КЛАВИАТУРЫ ЧЕРЕЗ БУФЕР (ФУНКЦИЯ 0AH)

Вызов: AH=0Ah

DS:DX - указатель на буфер ввода

Возвращает: --

Функция читает строку со стандартного ввода. DX должен содержать смещение (сегментный адрес в DS) буфера ввода, который имеет следующую структуру:

Байт	**Содержимое**
1	Максимальное количество символов в буфере, включая ВК (вы устанавливаете этот байт).
2	Количество введенных символов реально, без ВК (функция устанавливает этот байт).
3-n	Буфер, длиной не меньше величины, указанной в первом байте.

Символы читаются со стандартного ввода и помещаются в буфер, начиная с третьего байта. ENTER тоже читается (ВВОД). Когда буфер заполнен без одного символа, следующие символы не читаются, а на стандартный вывод посылается ASCII 07h (Звонок) до тех пор, пока не будет введен символ ВВОД. При вводе, строку можно редактировать.

ОПРОС СОСТОЯНИЯ КЛАВИАТУРЫ (ФУНКЦИЯ 0BH)

Вызов: AH=0Bh

Возвращает: AL=00h – если нет символов в буфере клавиатуры

AL=FFH – есть символы в буфере клавиатуры

Функция 0Bh проверяет, есть ли в буфере клавиатуры символы со стандартного ввода (если стандартный ввод не был переадресован, то опрашивается буфер клавиатуры). Если таковые есть, то AL возвращает FFh, в противном случае – 00h.

ОЧИСТИТЬ БУФЕР, ЧИТАТЬ С КЛАВИАТУРЫ (ФУНКЦИЯ 0CH)

Вызов: AH=0Ch

AL - 1,6,7,8 или 0AH = вызывается соответствующая функция

AL - любое другое значение = никаких последующих действий

Возвращает: AL=00h – буфер клавиатуры очищен и никаких последующих действий

Функция 0Ch очищает стандартный буфер ввода (если не переназначен, то буфер клавиатуры). Дальнейшие действия зависят от содержимого AL.

Регистр AL	**Действия**
1,6,7,8 или 0Ah	Вызывается соответствующая функция MS-DOS
Любое другое значение	Никаких последующих действий, AL возвращает 00h

ПОЛУЧИТЬ ВРЕМЯ (ФУНКЦИЯ 2CH)

Вызов: AH=2Ch

Возвращает: CH - часы (0-23)

CL - минуты (0-59)

DH - секунды (0-59)

DL - сотые доли секунды (0-99)

Функция 2Ch возвращает время операционной системы в регистрах CX и DX в двоичном коде.

В зависимости от аппаратной реализации отсчета времени, некоторые поля регистров не используются. Например, многие таймеры не выделяют долей секунды. В таких случаях значение DL, скорее всего, будет всегда равно 0.

Составитель Александр Викторович Карпов

СОЗДАНИЕ И ВЫПОЛНЕНИЕ ПРОСТЫХ ПРОГРАММ НА АССЕМБЛЕРЕ МП i8086 (часть I)

Методические указания к лабораторной работе