Лабораторная работа № 3

тема «Assembler. Адресация данных в памяти»

Выполните ассемблирование и компоновку программы, используя для этого следующие команды:

Текст программы:

```
1   IDEAL
2   MODEL small
3   STACK 256
4   DATASEG
5   exCode  DB     0
6   mem1 db 12h,34h,56h,78h
7   CODESEG
    …
11  push 9999h
12  push 0ffffh
13  ;прямая адресация
14  mov dh,[mem1]
15  ;переназначение
16  mov ch,[es:1]
17  mov ch,[cs:0]
18  mov ch,[ss:00FCh]
    …
```

1. Загрузите программу в Turbo Debugger.
2. Остановитесь в строке 14. Здесь приведена команда, использующая адрес смещения переменной в памяти относительно базового сегмента данных DS *(прямая адресация)*. Выполните эту команду и посмотрите, какое значение получит регистр DH в результате выполнения.
3. Обычно обращения к непосредственному адресу выполняются относительно сегмента данных DS. Чтобы это изменить, можно определить сегментное переназначение. Пример таких переназначений приведен в строках 16 - 18. В строке 16 в качестве базового сегмента используется дополнительный сегментный регистр ES. В результате выполнения этой команды в регистр DH будет загружен байт, расположенный со смещением 1 байт от адреса регистра ES.
4. Что произойдет в результате выполнения команд, находящихся в строках 17 и 18?
5. Вместо обращения к переменной в памяти по имени можно использовать один из трех регистров – BX, SI, DI в качестве указателя на данные в памяти. Поскольку для адресации различных областей памяти программа может изменять значения регистров, *косвенная адресация* позволяет одной командой оперировать многими переменными. Вставьте в текст программы 4 строки:

mov si, offset mem1 mov dx,[word si] mov di,sp
mov ch,[byte ss:di]

В первой строке осуществляется загрузка в регистр SI смещения переменной mem1. А во второй строке данные, адресуемые SI, пересылаются в 16-битовый регистр DX. В третьей и в четвертой строках байт данных, расположенных по адресу SS:DI, пересылаются в 8-битовый регистр CH. Предварительно в третьей строке в регистр DI загружается адрес стекового указателя SP.

Операторы Word и Byte необходимы, если Turbo Assembler не может определить, что адресует регистр в памяти – байт или слово. Во второй строке данные, адресуемые SI, пересылаются в 16-битовый регистр, следовательно в операторе Word нет необходимости.

6. Для *базовой адресации* используются два регистра – BX, BP. Обращение к BX проводятся относительно сегмента данных, адресуемого DS. Ссылки на BP осуществляются относительно стекового сегмента SS и обычно применяются для чтения и записи значений в стек. Для обращения к данным из любых других сегментов можно использовать переназначения.
7. Вставьте в текст программы следующие команды:

mov bx,offset mem1 mov bp, sp
mov ax,[bx+3] mov dx,[word bp+2]

Что произойдет в результате выполнения этих команд?

8. *Индексная адресация* аналогична базовой адресации за исключением того, что адреса смещений содержатся в регистрах SI и DI. До тех пор пока вы не определили сегментное переназначение, все обращения к индексным адресам проводятся относительно сегмента данных DS. Обычно индексная адресация используется для доступа к простым массивам.
9. Вставьте в текст программы следующие команды:

mov si, offset mem1
mov dx, [word si+2]

Что произойдет в результате выполнения этих команд?

10. *Базово-индексная адресация* объединяет два регистра и добавляет необязательные значения перемещения для формирования обращения к памяти по составному смещению – соединяя черты базового и индексного способов адресации. Первым регистром должен быть либо BX, либо BP. Вторым регистром должен быть либо SI, либо DI. Смещения, начинающиеся с BX, определяются относительно сегмента данных DS, смещения, начинающиеся с BP – относительно сегмента стека SS.

Примеры базово-индексной адресации:

```
mov ax, [bx+si]  ;загрузка в AX слова из сегмента данных
mov ax, [bp+di+3] ;загрузка в AX слова из сегмента стека
```

Можно использовать переназначения и ссылаться на данные в сегментах, отличных от установленных по умолчанию:

```
mov ax, [cs:bp+di]    ; используется CS вместо SS
mov ax, [es:bx+si+8] ; используется ES вместо DS
```

Определен массив из 24 байт:

```
SIM db "QWERTYUIOP{}"
    db "ЙЦУКЕНГШЩЗХЪ"
```

Загрузить в регистр DL элемент с индексом 6 из второго ряда, т.е. код ASCII ,буквы Г. Результат можно получить следующим образом:

```
mov bx, offset sim
mov si, 6
mov dl, [bx + si + 12]
```