

## **Лекция №6**

### **Тема " Turbo Debugger "**

#### **Turbo Debugger**

Работает в режиме супервизора, беря на себя управление программой и позволяя пошагово исполнять код программы и проверять значение переменных в памяти.

Можно выполнять программу до некоторой точки или до появления определенной ситуации.

Можно изменять значения в памяти, временно вставлять новые команды, а также менять значения регистров и флагов.

Для работы с Turbo Debugger необходимо выполнить ассемблирование и компоновку программы с опциями, которые добавляют отладочную информацию в OBJ- и EXE- файлы.

```
tasm /ZI Z3
```

```
tlink /V Z3
```

Загрузка Turbo Debugger:

```
td Z3
```

## Пример

Пример программы, изменяющей значения регистров (файл Z3.ASM).

```
1.  %Title "Использование команды MOV"
2.
3.  IDEAL
4.
5.  MODEL small
6.  STACK 256
7.
8.  DATASEG
9.  exCode DB 0
10. speed DB 99 ; Однобайтовая переменная
11.
12. CODESEG
13.
14. Start:
15. mov ax, @data ; Установка в DS адреса
16. mov ds, ax ; сегмента данных, т.к. в EXE- программах
17. ; переменные хранятся отдельно от кода программы
18.
19.     mov ax, 1 ; Переслать конкретные данные
20.     mov bx, 2 ; в регистры
21.     mov cx, 3
22.     mov dx, 4
23.
24.     mov ah, [speed] ; Загрузить значение speed в al
25.     mov si, offset speed ; Загрузить адрес speed в si
26.
27. Exit:
28. mov ah, 04Ch ; Функция DOS: выход из программы
29. mov al, [exCode] ; Возврат значения кода выхода
30. int 21h ; Вызов DOS. Остановка программы
31.
32. END Start ; Конец программы
```

**Лабораторная работа № 1****Тема: "Исполнение программы под управлением Turbo Debugger"**

Пример программы, изменяющей значения регистров (файл Z3.ASM).

```

1   %Title "Использование команды MOV"
2
3   IDEAL
4
5   MODEL small
6   STACK 256
7
8   DATASEG
9   exCode DB  0
10  speed DB  99   ; Однобайтовая переменная
11
12  CODESEG
13
14  Start:
15  mov ax, @data   ; Установка в DS адреса сегмента данных
16  mov ds, ax     ; т.к. в EXE- progr. переменные хранятся
17                ; отдельно от кода программы
18
19      mov ax, 1   ; Переслать конкретные данные
20      mov bx, 2   ; в регистры
21      mov cx, 3
22      mov dx, 4
23
24      mov ah, [speed] ; Загрузить значение speed в al
24      mov si, offset speed ; Загрузить в SI смещение speed относительно DS
26
27  Exit:
28  mov ah, 04Ch   ; Функция DOS: выход из программы
29  mov al, [exCode] ; Возврат значения кода выхода
30  int 21h       ; Вызов DOS. Остановка программы
31
32  END Start     ; Конец программы

```

Для работы с Turbo Debugger необходимо выполнить ассемблирование и компоновку программы с опциями, которые добавляют отладочную информацию в OBJ- и EXE- файлы.

```

tasm /ZI Z3
tlink /V Z3

```

Загрузите в Turbo Debugger файл Z3.EXE: **td Z3**

1. Для открытия окна CPU нажмите *Alt+V+C* (View/CPU/Windows – просмотр окна центрального процессора). Распахните его на весь экран, нажав *F5*. В окне CPU отображается информация – стек, регистры, флаги, память и команды.

2. После того как вы прочтаете следующие описания, нажмите *F8* для пошагового выполнения программы. Нумерация строк соответствует тексту программы *Z3.ASM*.
3. Строки 16-17 инициализируют сегментный регистр *DS*, занося в регистр предопределенное значение *@data* и пересылая его затем в регистр *DS*.
4. После выполнения строк 19-22 в регистры общего назначения *AX*, *BX*, *CX*, *DX* и записывают литерные значения 1, 2, 3, 4. Нажмите *F8*, пока стрелка, указывающая на исполняемые команды в *Turbo Debugger* (справа от адресов вида *CS:0011*), не остановится напротив команды *mov ah, [speed]*. Если вы случайно проскочили эту команду, нажмите для перезапуска программы *Ctrl+F2*, после чего снова нажмите *F8*, пока не вернетесь в нужное место.
5. Команда *mov ah, [speed]* в строке 24 загружает значение, содержащееся по адресу *speed*, в 8-битовый регистр *ah*. Обратите внимание на текст в двойной рамке в верхней части экрана – *DS:0001 = 63*. Он указывает на шестнадцатеричное значение *63*, которое будет загружено в регистр *ah*. Запись *DS:0001* отображает адрес, по которому было записано значение. Подобно другим адресам, он состоит из двух частей: значения сегмента (содержащегося в регистре *DS*) и смещения *0001*.
6. Для выполнения команды в строке 24 нажмите *F8* и наблюдайте в правом верхнем углу экрана за изменением значения в регистре *AX*. Обратите внимание, что запись *DS:0001=63* при этом исчезла. Чтобы снова увидеть ее, воспользуйтесь клавишами "стрелка вверх" и "стрелка вниз" для перемещения выделенного курсора. Перемещая прямоугольный курсор на любую отдельную команду, вы всегда можете наблюдать за результатами загрузки значений в регистры или память.
7. Найдите в правой верхней трети окна *CPU* регистр *SI*. Снова нажмите *F8*, выполнив команду *mov si, offset speed* в строке 25. Как вы видите, эта команда устанавливает *0001* в регистре *SI*, т.е. значение смещения адреса из предыдущего шага. Ключевое слово *OFFSET* в команде *mov* указывает, что в регистр *SI* загружается смещение *speed* относительно *DS*.
8. Продолжите нажимать *F8* до окончания программы. В строках 28-30 выполняются три шага, завершающие любую *EXE*-программу. Сначала в регистр *AH* загружается код функции выхода *DOS* (*04Ch*). Затем в регистр *AL* записывается содержимое переменной *exCode*, которая может возвращаться программой в *DOS* в качестве индикатора ошибки. Нулевое значение означает отсутствие ошибок. Команда *int 21h* в строке 30 вызывает *DOS* с параметрами, содержащимися в *AH* и *AL*.
9. Для выхода из *Turbo Debugger* нажмите *ESC* и *Alt+X*.

### Вопросы

1. Какие требуются шаги по ассемблированию и компоновке, чтобы создать необходимые файлы для отладки программы с помощью *Turbo Debugger*?
2. Как правильно заканчивать программу на языке ассемблер?
3. В чем состоит различие между ошибками и предупреждениями?
4. Какую программу, *Turbo Assembler* или *Turbo Linker*, вы используете для создания объектного кода? Какую программу необходимо использовать для создания исполняемого кода?
5. Для чего создается объектный код?