

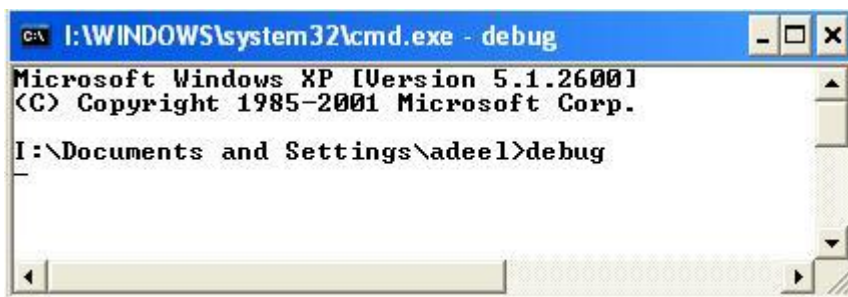
The term **debugging** refers to removal of **bugs** from your program. There are number of hardware and software tools, available for debugging.

The tool we will use for our assembly language programs is **DEBUG**, the one provided by the DOS. To use this utility we must know the commands, it provides, and how to run these commands. In this document you will find a brief description of frequently used commands.

## How to start DEBUG

In command line interface

I:\> debug ↵



**DEBUG** provides 24 commands.

To invoke any of these commands, enter the command character in either upper or lower case, and then enter any parameter that the command may require.

Commands are executed when you press enter, not as you type them. You can change the line by pressing the backspace key to delete what you have typed and then type the correct characters. When the command line is as you want it, then press enter to execute the entire line.

Debug does not detect errors until you have pressed Enter. If you make a syntax error in an entry, the command line is displayed with the word *error* added at the point at which the error was detected. If a command line contains more than one error, only the first error is detected and indicated, and execution ceases at that point.

In the following session we have a description for these commands.

## 1. Display Help Screen (?)

Once you are within DEBUG, using the question mark produces a screen of information that briefly recaps the syntax for DEBUG commands.

```
C:\ I:\WINDOWS\system32\cmd.exe - debug
-?
assemble      A [address]
compare        C range address
dump           D [range]
enter          E address [list]
fill           F range list
go             G [=address] [addresses]
hex            H value1 value2
input          I port
load           L [address] [drive] [firstsector] [number]
move           M range address
name           N [pathname] [arglist]
output         O port byte
proceed        P [=address] [number]
quit           Q
register        R [register]
search         S range list
trace          T [=address] [value]
unassemble     U [range]
write          W [address] [drive] [firstsector] [number]
allocate expanded memory  XA [#pages]
deallocate expanded memory XD [handle]
map expanded memory pages XM [Lpage] [Ppage] [handle]
display expanded memory status XS
```

## 2. Assemble (A)

The assemble command is used for entering assembly language instructions and for having them translated directly into the machine language instructions in memory. Its syntax is:

### -A *address*

*address* is an optional beginning address (in hexadecimal) at which the assembled machine language instructions are placed. If you do not specify an address, DEBUG starts placing the instructions either at **CS:0100** or after the last machine language instruction entered through Assemble (command).

```
C:\ I:\WINDOWS\system32\cmd.exe - debug
-a
0B33:0100 mov ax,34
0B33:0103 mov bx,43
0B33:0106 add ax,bx
0B33:0108
-
-
-a 200
0B33:0200 mov cx,23
0B33:0203 mov dx,43
0B33:0206 add cx,dx
0B33:0208
```

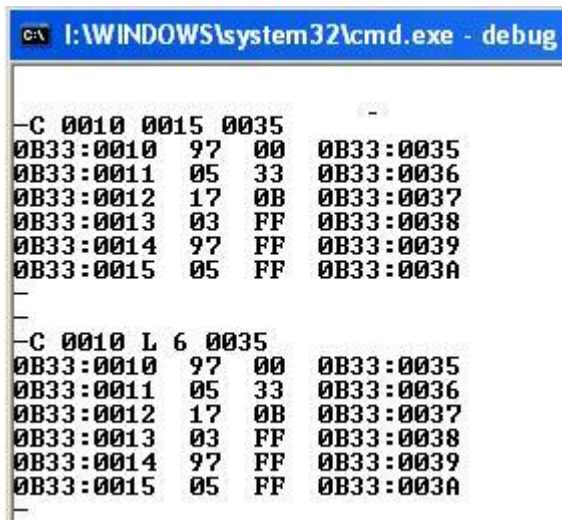
### 3. Compare (C)

The compare command compares and reports on any differences between the contents of two memory blocks. The syntax for this command is

#### **-C** *range address*

*range* is either both the beginning and ending addresses or, if preceded by an *L*, the beginning address and length of the first memory block. *address* is the start of the second memory block. The length of the second block is assumed to be equal to the length of the first. The memory blocks can overlap, and the second block can lie physically before the first.

The command compares the two blocks, byte-by-byte, and reports any differences in the following format:



```
C:\I:\WINDOWS\system32\cmd.exe - debug
-C 0010 0015 0035
0B33:0010 97 00 0B33:0035
0B33:0011 05 33 0B33:0036
0B33:0012 17 0B 0B33:0037
0B33:0013 03 FF 0B33:0038
0B33:0014 97 FF 0B33:0039
0B33:0015 05 FF 0B33:003A
-
-C 0010 L 6 0035
0B33:0010 97 00 0B33:0035
0B33:0011 05 33 0B33:0036
0B33:0012 17 0B 0B33:0037
0B33:0013 03 FF 0B33:0038
0B33:0014 97 FF 0B33:0039
0B33:0015 05 FF 0B33:003A
-
```

### 4. Dump (D)

The Dump command is one you will use often. It displays the contents of a series of memory locations. The syntax for this command is

#### **-D** *address1 address2*

You must specify *address1*, an optional starting address for the display, before you can specify *address2*, an optional ending address.

If no addresses are specified, DEBUG starts displaying memory locations with **DS:0100** or, (if Dump already has been used) with the byte following the last byte displayed by the most recent Dump command.

Dump always displays 16 bytes per line, beginning with the nearest paragraph (16-byte) boundary. This display rule may differ with the first and last lines displayed, because you may have asked DEBUG to start the dump with a memory location that was not on a paragraph boundary.

If you do not specify an ending address, DEBUG always displays 128 bytes of memory. Each byte is shown in both Hexadecimal and ASCII representation.

```

C:\ I:\WINDOWS\system32\cmd.exe - debug
I:\DOCUMENT1\adeel>debug
-
-d
0B33:0100 B8 34 00 BB 43 00 01 D8-99 80 3E 20 99 00 75 24 .4..C.....> ..u$
0B33:0110 A2 24 99 0A C9 75 1D 0A-C0 74 19 8B 34 00 22 0B .$. ..u...t..4."
0B33:0120 13 B0 1A 06 33 FF 8E 06-00 96 F2 AE 07 75 05 4F ....3.....u.0
0B33:0130 89 3E 21 96 BB 06 97 80-3E 13 96 00 74 03 BB 4C .>!.....>...t..L
0B33:0140 97 BE C3 96 8B 3E 05 98-B9 08 00 AC 3C 3F 75 02 .....>.....<?u.
0B33:0150 8A 07 3C 20 74 01 AA 43-E2 F1 B1 03 B0 20 38 04 ..< t..C..... 8.
0B33:0160 74 12 B0 2E AA AC 3C 3F-75 02 8A 07 3C 20 74 01 t.....<?u...< t.
0B33:0170 AA 43 E2 F1 32 C0 AA C3-F6 46 04 02 75 43 8B D5 .C..2....F..uC..
-
-d 0010 001F
0B33:0010 97 05 17 03 97 05 5F 04-01 01 01 00 02 FF FF FF ....._.....
-
-d 0010 L 10
0B33:0010 97 05 17 03 97 05 5F 04-01 01 01 00 02 FF FF FF ....._.....
-

```

## 5. Enter (E)

Enter command enables you to change the contents of specific memory locations. The syntax for this command is

### -E *address changes*

*address* is the beginning address for entering changes, and *changes* is an optional list of the changes to be made.

You can specify *changes* on the command line. But if you do not specify *changes* on the command line, DEBUG enters a special entry mode in which the values of memory locations, beginning at *address*, are displayed. You can change these values one byte at a time. (Be sure to enter the changes as hexadecimal numbers) After each entry, press the space bar to effect the change. The next byte is then displayed so that you can make any necessary changes. To exit entry mode and return to DEBUG command mode, press Enter.

If you enter a minus sign or hyphen as part of a change to a byte, DEBUG goes back one byte to the preceding memory location. You can then make additional changes to that byte.

If you have not made any changes to a byte, press the space bar to proceed to the next byte. The unchanged byte retains its original value.

```

C:\ I:\WINDOWS\system32\cmd.exe - debug
I:\DOCUMENT1\adeel>debug
-
-E 0010
0B33:0010 97.D3 05.89 17. 03.56
-
-

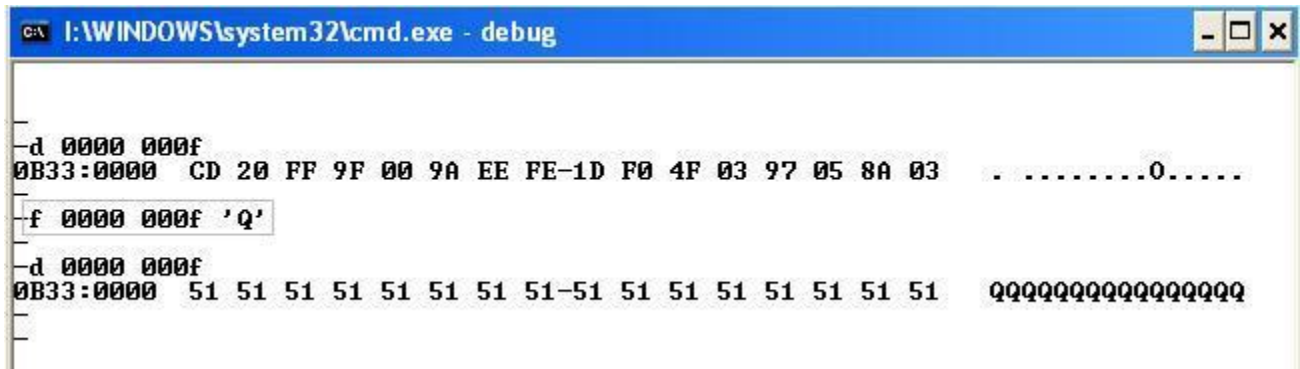
```

## 6. Fill (F)

This command is used to fill a block of memory with a specific value or series of values. The syntax for this command is

### **-F** *range fillvalue*

*range* is either both the beginning and ending addresses or, if preceded by an L, the beginning address and length of the memory block. *fillvalue* is the byte value that should be used to fill the memory block. If *fillvalue* represents fewer bytes than are needed to fill the *range*, the series is repeated until the *range* is completed.



```
C:\ I:\WINDOWS\system32\cmd.exe - debug
-d 0000 000f
0B33:0000  CD 20 FF 9F 00 9A EE FE-1D F0 4F 03 97 05 8A 03  . . . . .0. . . .
-f 0000 000f 'Q'
-d 0000 000f
0B33:0000  51 51 51 51 51 51 51 51-51 51 51 51 51 51 51  QQQQQQQQQQQQQQQQ
```

## 7. GO (G)

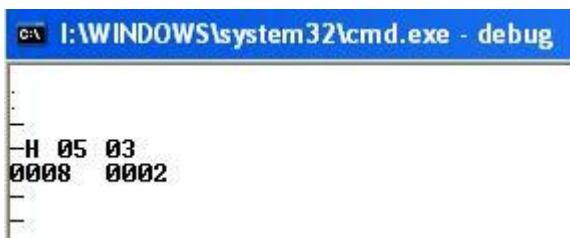
The GO command causes machine language statements to be executed. If you are debugging a program, this command executes the program you have loaded.

## 8. Hexadecimal Arithmetic (H)

This command does simple hexadecimal addition and subtraction. The syntax for this command is

### **-H** *value1 value2*

*value1* and *value2* are hexadecimal numbers. This command returns a result line that shows two values: the sum of *value1* and *value2*, and the difference between *value1* and *value2*. This command does not alter any registers or flags.



```
C:\ I:\WINDOWS\system32\cmd.exe - debug
-H 05 03
0008  0002
```

## 9. Input (I)

The input command fetches a byte from a port. The syntax is

### **-I** *port*

*port* is address of the specified port to read.

## 10. Load (L)

This command is used to load a file in memory.

## 11. Move (M)

The move command moves a block of memory from one location to another. The syntax is

**-M *range address***

*range* is either both the beginning and ending addresses or, if specified by an *L*, the beginning address and the length of the first memory block. *address* is the destination address for the move. The destination address and source block can overlap. The bytes from the source block are moved, one at a time, to the destination address.

## 12. Name (N)

The Name command is used to specify a file name to be used either by the Load or Write commands or by the program you are debugging. The syntax for this command is

**-N filename**

## 13. Output (O)

The Output command outputs a byte to a specified port. The syntax is

**-O *port value***

*port* is the address of the specified port, and *value* is the hexadecimal byte to write.

## 14. Proceed (P), Trace (T)

Using this command, you can execute machine language instruction in a single step, after which the register status is displayed. The syntax is

**-P**

OR

**-T**

The **P** command differs from **T** only in its handling of the CALL and INT instructions; **P** executes the entire called routine before pausing, whereas **T** executes only the transfer of control and then pauses at the first instruction of the called routine.

## 15. Quit (Q)

The Quit command is used to quit DEBUG and return control of the computer to DOS. The syntax is

**-Q**

## 16. Register (R)

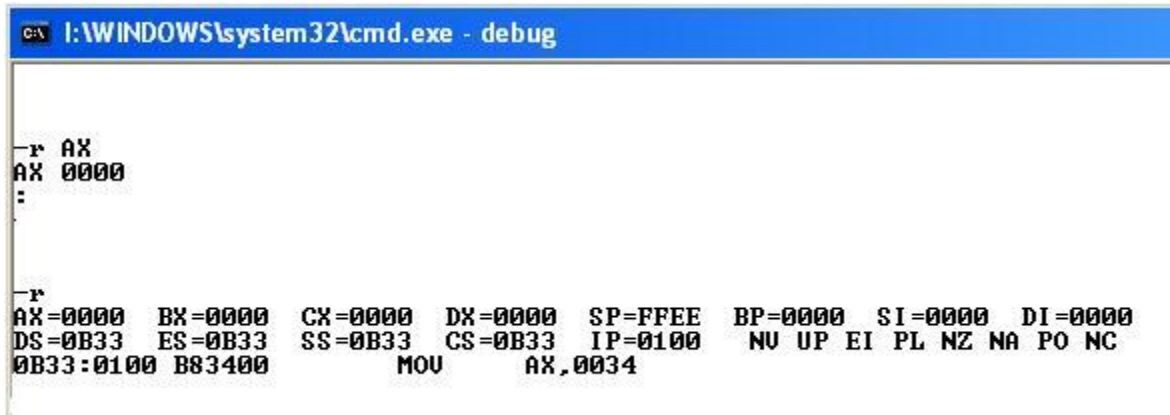
The Register command displays the microprocessor's register and flag values, and enables you to change individual register values. The command's syntax is

**-R *register***

*register*, the optional name of the register to modify, may be any of the following: AX, BX, CX, DX, SP, BP, SI, DI, IP or F. F refers to the flag register.

If you enter the Register command with no parameters, DEBUG responds by displaying a register summary.

If you enter a register name as a parameter, DEBUG displays the current register value and waits for you to enter a new value. If you enter a value, it is assumed to be in hexadecimal. If you do not enter a value, no change is made to the register value.



```
C:\ I:\WINDOWS\system32\cmd.exe - debug

R AX
AX 0000
:

R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B33 ES=0B33 SS=0B33 CS=0B33 IP=0100  NU UP EI PL NZ NA PO NC
0B33:0100 B83400      MOV     AX,0034
```

## 17. Search (S)

With this command, you can search a block of memory for a specific sequence of values. The syntax is

**-S range searchvalue**

*range* is either both the beginning and ending addresses or, if specified by an *L*, the beginning address and the length of the first memory block. *searchvalue* is the byte value you want to search for in the memory block.

The values that DEBUG searches for can be any combination of hexadecimal numbers and ASCII characters. ASCII characters must be enclosed in quotation marks.

If DEBUG locates any exact matches, it displays the address of the beginning of the match. If no matches are found, no message is displayed.

## 18. Unassemble (U)

The Unassemble command decodes the values of a group of memory locations into 8088 mnemonics. One of the most frequently used DEBUG commands, Unassemble enables you to view the instructions that are executed during the DEBUG operation. The syntax is as follows:

**-U address range**

*address*, which is optional, is the beginning address of the area to be unassembled. *range*, which is optional if *address* is specified, is either the ending address of the area or, if preceded by an *L*, the length of the area. If you specify an *address* but no *range*, approximately one screenful of data is displayed.

If you do not specify an *address* or *range*, unassembly begins with the memory location indicated by **CS:IP** or (if Unassemble has already been used) with the byte following the last byte displayed by the most recent Unassemble command. The unassembly proceeds for 16 bytes. The number of instruction lines this process represents depends on the number of bytes used in each instruction line. If you specify an *address* and a *range*, all bytes within that block are unassembled.

**Note:**

There is one more utility provided by windows to perform all of above defined operations. It is CodeView.

The syntax to start it is as follows

```
C:\> CV filename.exe |  
    ↙
```

Explore it by yourself.