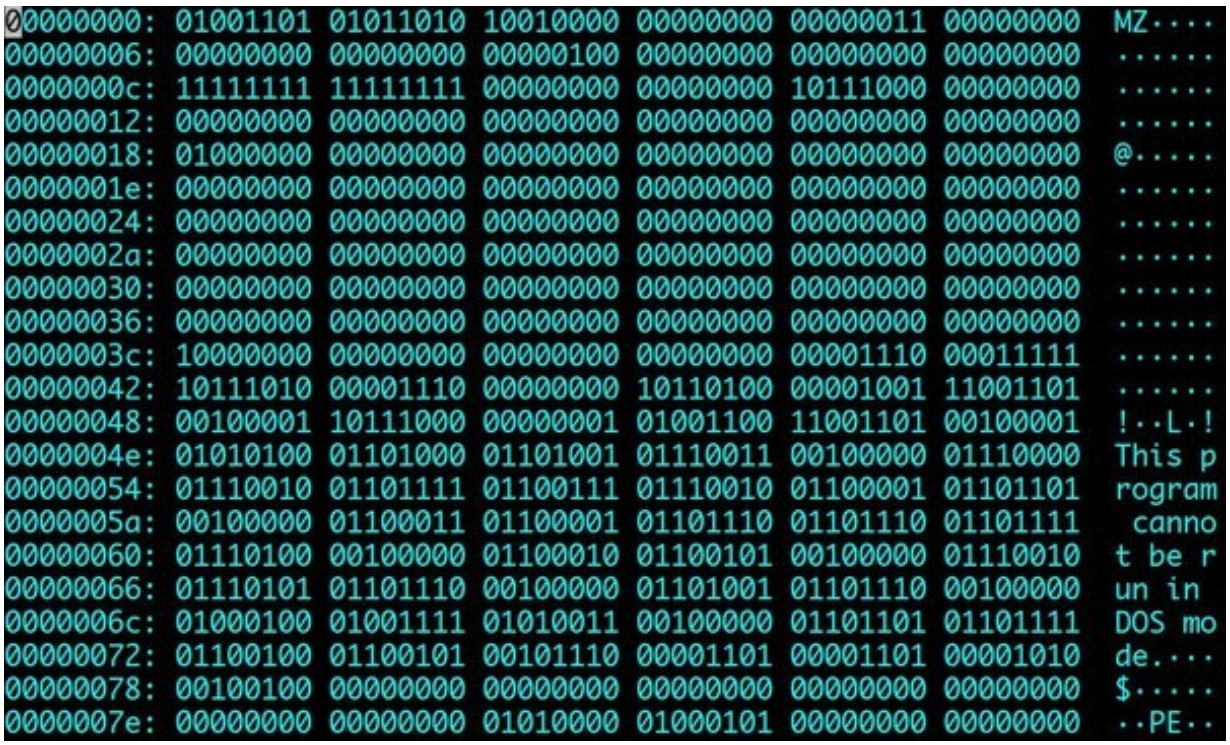
Computer Organization
And
Assembly Language



Lab Manual 01, 02

Objectives:

1. Understanding Debugger
2. Understanding different commands of debugger
3. Assembling, Unassembling and Tracing simple codes using debugger

Lab Instructor:

Mr. Tariq Mehmood Butt tariq.butt@pucit.edu.pk

Teacher Assistants:

Hafiz Muhammad Ahmad	bcsf21m502@pucit.edu.pk
Syed Muhammad Zain Raza	bcsf21m510@pucit.edu.pk
Zahra Malik	bcsf21m551@pucit.edu.pk
Bilal	bsdsf21m022@pucit.edu.pk

What is Debugger?

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↵

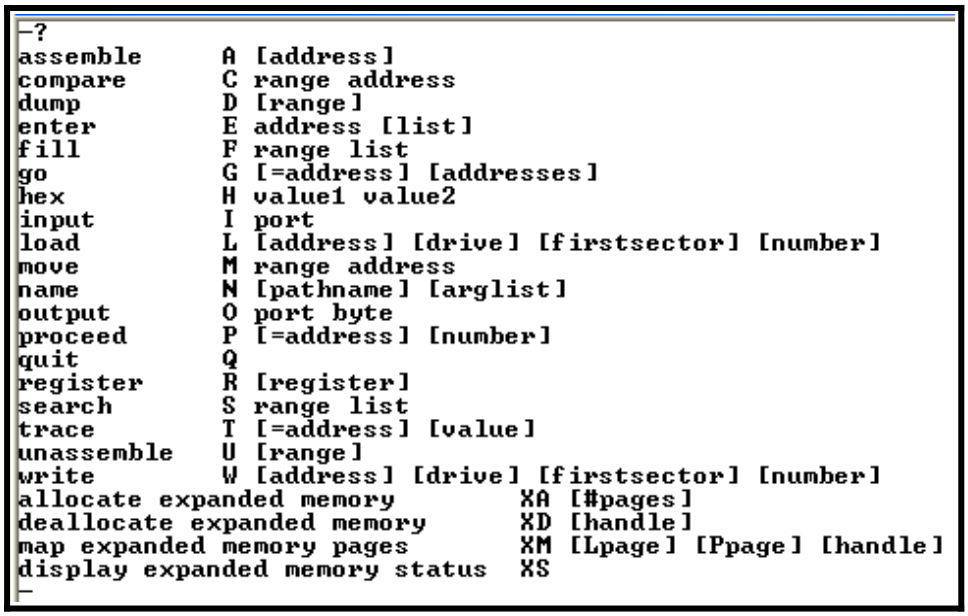
DEBUG provides **24** commands.

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 theline 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 firsterror 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.



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).

```
-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 as in the given format:

```
-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

```
I:\DOCUME~1\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.O
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   ......_.........
-
-
```

with DS:0100 or, (if Dump already has been used) with the byte following the last byte displayed by themost 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.

5. Enter (E)

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

-E *address changes*

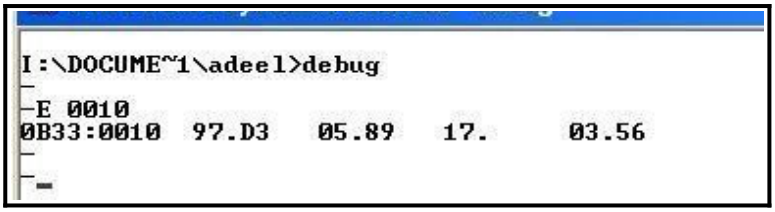
address is the beginning address for entering changes, and changes is an optional list

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.



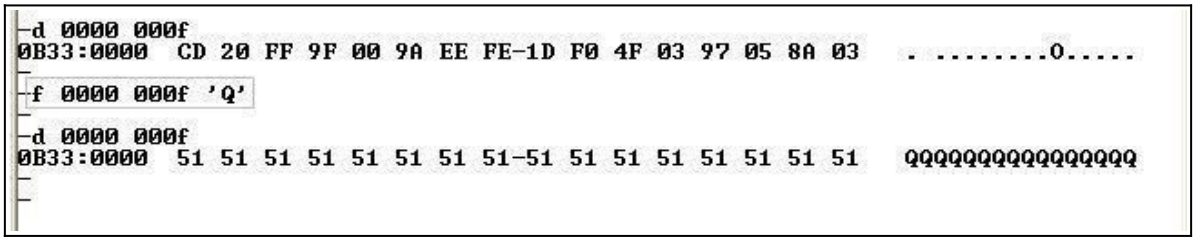
6. Fill (F)

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 fillthe range, the series is repeated until the range is completed.

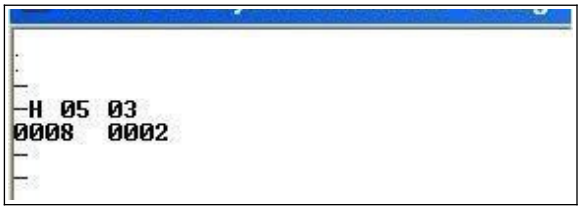


7. GO (G)

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.

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, afterwhich 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

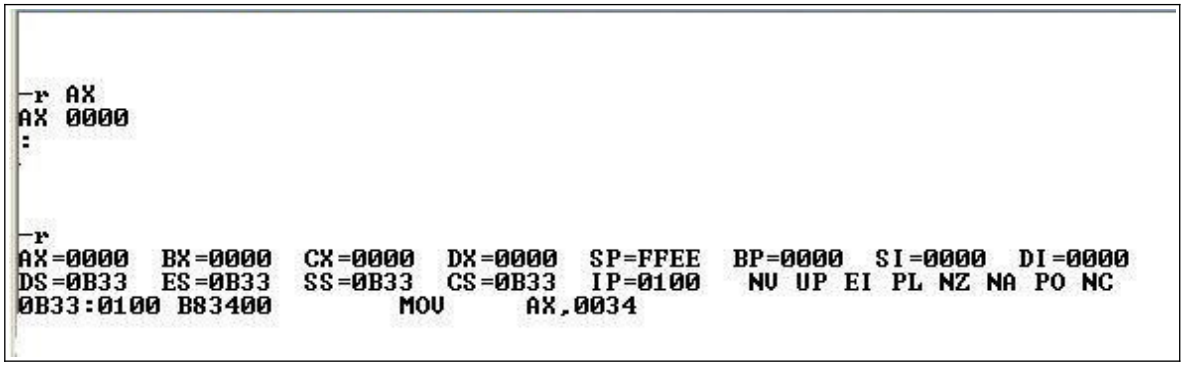
-R *register*

register, the optional name of the register to modify, may be any of the *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 aregister 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.



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 8086 mnemonics. One of the most frequently used DEBUG commands, Unassemble enables you to view the instructions that are executed during the DEBUG operation. The syntaxis 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.

Lab Tasks

Note: Use only debug commands for the following tasks.

Task 1:Write a sequence of commands to change the current value of ES register to 0EE.

Task 2:Write a sequence of commands to display the data on memory from 0100 to 80h bytes.

Task 3: Write a sequence of command to

a) Enter string ‘Hello’ in memory starting at 2

b) Display just the message ‘Hello’ that you have enter into the memory in part a.

Task 4: What will the following command do?

-U CS:100 1E0

Task 5: Which register refer to code? Just write answer.

Task 6: Which command is used to exit from debugger?

Task 7: Run the following codes on debugger (using assemble command) and write down the status of flags:

i)

```
Mov ax,FF12
Mov bx,0012
Add ax,bx
```

ii)

```
Mov al,0001
Dec al
```

iii)

```
Mov al,ff
Inc al
```

iv)

```
Mov ax,40
Mov bx,50
Sub ax,bx
```

Task 8: Write assembly instructions which should:

- Change the contents of AX to 1234
- Add 1 to the contents of AX
- Copy the value of AX to DX
- Subtract 1233 from the value of DX
- Do BH = DL
- Place 9 in AL

Task 9: Write assembly instructions which should:

- Change the contents of AX to 4000H
- Add AX to AX
- Subtract 0FFFFH from AX
- Increment in AX
- Decrement in AX

Task 10: Write assembly instructions that exchanges the values of AX and BX.

(You can take values of your choice. Please note down your assembly instruction also.)

Task 11: Write an assembly code in debugger using assemble (a) command which can copy the contents of an 8-byte array (memory) from 0100 – 0107 offset having segment 4000 to the array located on offset 0200 – 0207 in the same segment (4000). Use E (enter) command for initializing the source memory (0100-0107) with some data.

Task 12: Write an assembly code in debugger using assemble (a) command which can copy the contents of an 8-byte array (memory) from 0100 – 0107 offset having segment 4000 in reverse order to the array located on offset 0200 – 0207 in the same segment (4000). Use E (enter) command for initializing the source memory (0100-0107) with some data.

Task 13: Write an assembly code in debugger using assemble (a) command which can SWAP the contents of an 8-byte array from 0100 – 0107 offset of segment 4000 with the contents of an 8-byte array on offset 0200 – 0207 in the same segment (4000). Use E (enter) command for initializing the both arrays with some data.

Task 14: Write an assembly code in debugger using assemble (a) command which can Reverse SWAP the content of 8-byte arrays (memory) from 0100 – 0107 offset of segment 4000 with the memory located on offset 0200 – 0207 of same segment (4000). Use E (enter) command for initializing the both arrays (memory) with some data. Note: Verify your code by execution using trace (t) command. Also check the contents of memory using dump (d) command.