Laboratory Manual

For

CS/EEE/INSTR/ECE F241
Microprocessor Programming & Interfacing

Birla Institute of Technology & Science, Pilani, Pilani Campus

Lab2- Introduction to DebugX

1.0 Introduction to DebugX

After completing the previous lab session, hope you are more confident with using DebugX. In this lab session, we start with a recap of using DebugX and an overview of the commands. This is followed by a more detailed exploration of the commands. Finally, you will be exposed to a few other commands in the tasks for this session.

1.1 How to start DebugX

- Click on the DOSBox icon
- At the command prompt, type mount d: d:\masm611\bin” [Replace ‘d’ with the drive containing the folder MASM611]
- Again type d:
- Finally, type ‘debugX’

1.2 Rules of DebugX

- It does not distinguish between lower case and upper case letters.
- It assumes that all numbers are in Hexadecimal unlike Microsoft assembler (MASM, which you will use in upcoming labs), which assumes numbers to be decimal.
- Segment and offset are specified as segment: offset.

1.3 Some DebugX Commands

DebugX Commands [can be typed on debugX terminal]

A	Assembles symbolic instructions into machine code
D	Display the contents of an area of memory in hex format
E	Enter data into memory beginning at specific location
G	Run the executable program in the memory (Go)
P	Proceed, Execute a set of related instructions
Q	Quit the debug session
R	Display the contents of one or more registers in hex format
T	Trace the execution of on instruction
U	Unassemble machine code into symbolic code
?	Debug Help to get a list of available commands.

1.4 R, the Register command: examining and altering the content of registers

In DEBUGX, the register command “R” allows you to examine and/or alter the contents of the internal CPU registers.

R <register name>

The “R” command will display the contents of all registers unless the optional <register name> field is entered. In which case, only the content of the selected register will be displayed. The R command also displays the instruction pointed to by the IP. The “R” command without the register name field, responds with three lines of information. The first two lines show you the programmer’s model of the 80x86 microprocessor. (The first line displays the contents of the general-purpose, pointer, and index registers. The second line displays the current values of the segment registers, the instruction pointer, and the flag register bits. The third line shows the location, machine code, and Assembly code of the next instruction to be executed.)

If the optional <register name> field is specified in the “R” command, DEBUGX will display the contents of the selected register and give you an opportunity to change its value. Just type a <return> if no change is needed, e.g.

```
- r ECX

ECX 00000000

: FFFF

- r ECX

ECX 0000FFFF

:
```

Note that DEBUGX pads input numbers on the left with zeros if fewer than four digits are typed in for 16-bit register mode and if fewer than eight digits in 32 bit addressing mode

1.5 A, the Assemble command

The assemble command is used to enter Assembly language instructions into memory.

A <starting address>

The starting address may be given as an offset number, in which case it is assumed to be an offset into the code segment. Otherwise, CS can also be specified explicitly. (eg. “A 100” and “A CS:100” will achieve the same result). When this command is entered, DEBUG will begin prompting you to enter Assembly language instructions. After an instruction is typed in and followed by <return>, DEBUG will prompt for the next instruction. This process is repeated until you type a <return> at the address prompt, at which time DEBUG will return you to the debug command prompt.

Use the “A” command to enter the following instructions starting from offset 0100H.

```
32-bit format                                16-bit format

A 100                                        - A 100

xxxx:0100  mov eax,1                         xxxx:0100  mov ax,1

xxxx:0106  mov ebx,2                         xxxx:0103  mov bx,2

xxxx:010C  mov ecx,3                         xxxx:0106  mov cx,3

xxxx:0112  add eax, ecx                      xxxx:0109  add ax, cx

xxxx:0115  add eax, ebx                      xxxx:010B  add ax, bx

xxxx:0118   jmp 100                          xxxx:010D   jmp 100

xxxx:011A <enter>                            xxxx:010F <enter>
```

Where xxxx specifies the base value of instruction address in the code segment. Please note that the second instruction starts at xxxx:0106 in 32 bit format and xxxx:0103 in 16-bit format. This implies that first instruction is six bytes long in 32-bit format and three byte long in 16-bit format.

Similarly note the size of all instructions.

When DEBUG is first invoked, what are the values in the general-purpose registers?

1.5 U, the Unassemble command: looking at machine code

The unassemble command displays the machine code in memory along with their equivalent Assembly language instructions. The command can be given in either format shown below:

U <starting address> <ending address>

```
32-bit format                                  16-bit format

U 100 118                                      U 100 10D

xxxx:0100  66B801000000 mov eax,01             xxxx:0100  B80100  mov  ax,1

xxxx:0106  66BB02000000 mov ebx,02             xxxx:0103  BB0200  mov  bx,2

xxxx:010C  66B903000000 mov ecx,03             xxxx:0106  B90300  mov  cx,3

xxxx:0112  6603C3 add eax, ebx                 xxxx:0109   03C3    add   ax,bx

xxxx:0115  6603C1 add eax, ecx                 xxxx:010B   03C1    add   ax,cx

xxx:0118    EBE6 jmp 0100                      xxxx:010D   EBF1     jmp 100
```

All the data transfer and arithmetic instructions in 32-bit format has 66 as a prefix why?

1.6 E, the Enter Command

E address

You can modify the content of any memory location in the data segment using the "E" command

1.7 T, the Trace command: single-step execution

The trace command allows you to trace through the execution of your programs one or more instructions at a time to verify the effect of the programs on registers and/or data.

To execute one instruction, use command 'T'

To execute more than one instruction:

T < =starting address> < number of instructions>

T =100 5

If you do not specify the starting address then you have to set the IP using the R command before using the T command

Usage

Using r command to set IP to 100 and then execute T 5

Using r command to set IP to 100 and then execute T

Trace through the above sequence of instructions that you have entered and examine the registers and Flag registers

1.8 G, the Go command

The go command instructs DEBUG to execute the instructions found between the starting and stop addresses.

G < = starting address > < stop address >

Execute the following command and explain the result

G = 100 118

Caution: the command G= 100 119 will cause the system to hang. Explain why?

Often, it is convenient to pause the program after executing a few instructions, thus effectively creating a break point in the program. For example, the command "g 10c" tells the DEBUG to start executing the next instruction(s) and pause at offset 0106H.

The main difference between the GO and TRACE commands is that the GO command lists the register values after the execution of the last instruction while the TRACE command does so after each instruction execution

Try executing the sequence of instructions you entered using several options of the "G" command. Remember to reload 0100 into IP every time you use G without the staring address.

Tasks to be completed

Task1: To search for a string in a memory location.

1. Load a txt file having a string ‘virus’ anywhere in the file using N and L commands of debug.

‘N’ Names a program or a file you intend to read or write onto disk.

Format of command N path:name.ext

If the file is in MASM611\bin folder and the name of the file is f1.txt use command
N f1.txt

2. Load File into memory using L command.

‘L’ Loads a File or Disk Sectors into memory.

L 100 will load the file specified by N into CS:100.

3. Search for string ‘virus’ using S command.

‘S’ Searches memory for a string.

Format S start-addr L val “data”

Format S start-addr end-addr “data”

4. Edit the contents of the file in memory as follows

Repace the “virus” in file with abcde”

[Note: Using W command to update the file]

Format W

Task2: To use debug to create a very small machine language program that you save on disk as a file.

1. Store data in DS:0200 using E command.

2. Using A command write a small ALP that will transfer the contents of this memory location into AX,BX,CX,DX,SI,DI registers.

3. Use N command to create a file. (Note: The executable file that you will create will have .com extension)

4. Enter value ‘0’ in BX and size of program in ‘CX’

5. Using W command now write executable program.

Format W

6. Exit Debug.

7. Open Debug with ‘name of file’.com and trace through the program and examine contents of registers.

Task 3: Create a .txt file that contains your First Name followed by “MUP,” separated by a ‘ ✱.’ For example, if your name is ‘Apple,’ then the content of the file should be

Apple*MUP

Modify the contents of the file as follows using the steps for file modification covered in this lab.

Apple[MUP]

Note:

Character	ASCII Value
[	5B
]	5D
*	2A