Computer Science

Level 2

Section #2

Assembly Programming

2020

Section #2 -- Debug Commands

Q.1 What is register?

- Registers used to Store information temporarily in CPU (1 or 2 byte).
- **General-purpose registers** can be accessed as 16-bit or 8-bit registers.
- **All other registers** can be accessed only as the full 16 bits.
- The data always fill the low 8-bits of a register then if still needed use some or all of the high 8-bits.

Category	Bits	Register
General	16	AX, BX, CX, DX
	8	AH, AL, BH, BL, CH, CL, DH, DL
Pointer	16	SP (Stack Pointer), BP (Base Pointer).
Index	16	SI (Source Index), DL (Destination Index)
Segment	16	CS (Code segment), DS (Data Segment) SS (Stack Segment), ES (Extra Segment).
Instruction	16	IP (Instruction Pointer).
Flag	16	FR (Flag Register).

Q.2 What is debug?

- Is a DOS command works directly with memory and Processor registers, used for programming in assembly language.
- Can only create files with .COM extension.
- The size cannot be larger than 64Kb.
- You can see all debug command by write ? in DOS.
- All number in hexadecimal.
- Can use upper and lower case.
- Ctrl-c stop any command.

❖ Debug Commands

1) A *Assemble*

» Used to enter assembly language.

```
DOSBox 0.74, Cpu speed:   3000 cy

C:\>debug
-a100
073F:0100 mov ax,1
073F:0103 mov bx,2
073F:0106 mov cx,3
073F:0109 add ax,bx
073F:010B add ax,cx
073F:010D int 3
073F:010E
```

2) U UnAssemble

» Can be written in two formats [U StartA EndA, U StartA L#bytes].

» You cannot assemble lower than 100 because the beginning 256 bytes are reserved by DOS.

```
-u 100 10d
073F:0100 B80100        MOV     AX,0001
073F:0103 BB0200        MOV     BX,0002
073F:0106 B90300        MOV     CX,0003
073F:0109 01D8          ADD     AX,BX
073F:010B 01C8          ADD     AX,CX
073F:010D CC            INT     3
```

3) G Go

» Execute instruction between two addresses.

» If no address begin execute instruction at CS:IP until a breakpoint is reached.

» INT 3 terminate execution.

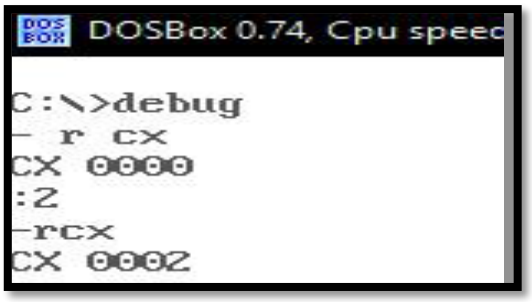
```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:   DEBUG

C:\>debug
-a100
073F:0100 mov ax,2
073F:0103 mov bx,3
073F:0106 add ax,bx
073F:0108 int 3
073F:0109
-g

AX=0005  BX=0003  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0108   NV UP EI PL NZ NA PE NC
073F:0108 CC             INT     3
-g=100 103

AX=0002  BX=0003  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PE NC
073F:0103 BB0300        MOV     BX,0003
-g=100

AX=0005  BX=0003  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0108   NV UP EI PL NZ NA PE NC
073F:0108 CC             INT     3
-
```

4) Q Quit

- Exit from debug program.

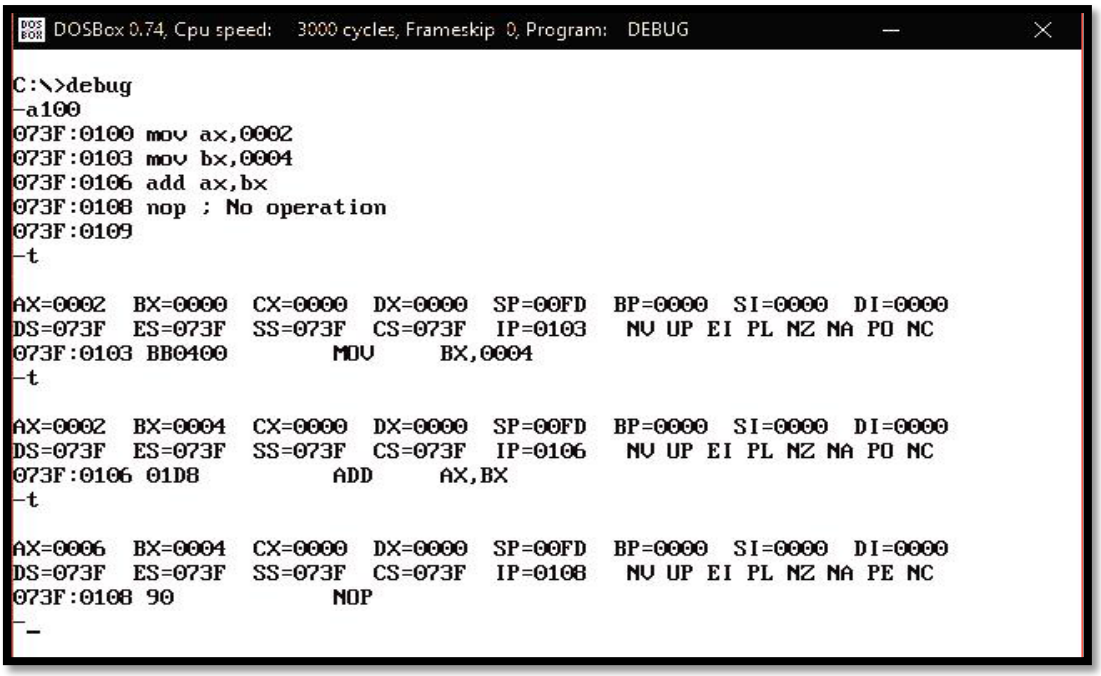
5) R Registers

- Display and modifying registers.
- R CX
  Display only contents of CX register followed by :
- You can enter hex number to change the content of cx register.
- If the hex fewer than four digits, will pad on left with zero.
- You cannot insert digit less than 0000 and larger than FFFF.



6) T Trace

- Trace the contents of one instruction.
- Can write in form [T begin address #of Instruction].



```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:   DEBUG
-a100
073F:0100 mov ax,0002
073F:0103 mov bx,0004
073F:0106 add ax,bx
073F:0108 mov cx,ax
073F:010A
-t=100 4

AX=0002  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 BB0400        MOV     BX,0004

AX=0002  BX=0004  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0106   NV UP EI PL NZ NA PO NC
073F:0106 01D8          ADD     AX,BX

AX=0006  BX=0004  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0108   NV UP EI PL NZ NA PE NC
073F:0108 89C1          MOV     CX,AX

AX=0006  BX=0004  CX=0006  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=010A   NV UP EI PL NZ NA PE NC
073F:010A D801                  FADD    DWORD PTR [BX+DI]             DS:0004=00

-
```

True or False:

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frames
C:\>debug
-a100
073F:0100 mov bh,2
073F:0102 mov ax,bh
                 ^ Error
073F:0102 mov ah,123
                   ^ Error
073F:0102 mov ax,12345
                    ^ Error
073F:0102 mov bx,0003
073F:0105 mov al,bx
                 ^ Error
073F:0105 mov dh,02
073F:0107 mov sl,dh
               ^ Error
073F:0107
```

7) F Fill

- Filling block of memory with data.
- [F StartA EndA Data]

```
DOSBox 0.74, Cpu speed:   30
C:\>debug
-f 100 10f ff 16 byte
-f cs:100 1ff 20
-      256 byte
```

8) D *Dumb*

- Display the contents of an area of memory.
- We can write Dumb command in three formats:

D [from] e.g: D 100	D [from] [to] e.g: D 100 130	D [address] [length] e.g: D 100 L3

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:   DEBUG

C:\>debug
-f 100 14f 20
-f 150 19f 00
-d 100 19f
073F:0100  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
073F:0110  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
073F:0120  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
073F:0130  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
073F:0140  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
073F:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0180  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0190  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-_
```

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:   DEBUG

C:\>debug
-f cs:100 12f 20
-d cs:100 12f
073F:0100  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
073F:0110  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
073F:0120  20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
```

9) E *Enter*

- Enter data into memory, beginning at a specific location.

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:   DEBUG

C:\>debug
-e200 "Welcome to met$"
-d200
073F:0200  57 65 6C 63 6F 6D 65 20-74 6F 20 6D 65 74 24 00   Welcome to met$.
073F:0210  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0220  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0230  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0240  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-_
```

» You can alter data with E

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:   DEBUG
-d200
073F:0200  57 65 6C 63 6F 6D 65 20-74 6F 20 6D 65 74 24 00   Welcome to met$.
073F:0210  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0220  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0230  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0240  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-e203
073F:0203  63.66

-e203
073F:0203  66.66   6F.67   6D.68   65.69

-d200
073F:0200  57 65 6C 66 67 68 69 20-74 6F 20 6D 65 74 24 00   Welfghi to met$.
073F:0210  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0220  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0230  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0240  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

» Example to enter data with E and execute instruction on it.

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:   DEBUG

C:\>debug
-e dS:0200 10 20 30 40 50
-d ds:0200 020f
073F:0200  10 20 30 40 50 68 69 20-74 6F 20 6D 65 74 24 00   . 0@Phi to met$.
-a100
073F:0100 mov al,00
073F:0102 add al,[0200]
073F:0106 add al,[0201]
073F:010A add al,[0202]
073F:010E add al,[0203]
073F:0112 add al,[0204]
073F:0116 int 3
073F:0117
-g=100 116

AX=00F0  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0116   NV UP EI NG NZ NA PE NC
073F:0116 CC            INT     3
-
```

» Example for moving data between memory and register.

```
C:\>debug
-e DS:0200 70
-d DS:0200 L1
073F:0200  70                                                    p
-a100
073F:0100 mov bx,[0200]
073F:0104 int 3
073F:0105
-t

AX=0000  BX=0070  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0104   NV UP EI PL NZ NA PO NC
073F:0104 CC             INT     3
-a100
073F:0100 mov ax,1234
073F:0103 mov [0200],ax
073F:0106 int 3
073F:0107
-g=100 107

AX=1234  BX=0070  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0106   NV UP EI PL NZ NA PO NC
073F:0106 CC             INT     3
-d DS:200 L1
073F:0200  34                                                    4
-d DS:0200 L2
073F:0200  34 12                                                 4.
```

10) H Hex

» Add and subtract two hex values.

» Show sum first, then difference.

» E.g:

-h aaa 531 0FDB 0579	-h fff 3 1002 0FFC	-h dbf ace 188D 02F1
-h 4 fffc 0000 0008	-h 100 123 0223 FFDD	-h 7FFF 8000 FFFF FFFF

» » -1 >> FFFF How?

0001

0001 0000 0000 0000

1111 1111 1111 1111

F F F F

11) M Move

- Move or copy data from one location to another.
- M [StartA EndA DestinationA].
- M 130 Lf 140

12) C Compare

- Used to check two areas of memory and display bytes that contain different data.
- If two area are identical, debug replay with -
- C [StartA EndA StartCompareA].
- C 130 L5 140

13) S Search

- Used to search a block of data for a specific value.
- S[StartA EndA Value]
- S 150 12f ff
- Example for M, C, S

```
-f 130 13f ff
-d 130 15f
073F:0130  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
073F:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
073F:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-m 130 13f 140
-d 130 15f
073F:0130  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
073F:0140  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF   ................
073F:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-c 130 134 140
-c 130 134 150
073F:0130  FF  00  073F:0150
073F:0131  FF  00  073F:0151
073F:0132  FF  00  073F:0152
073F:0133  FF  00  073F:0153
073F:0134  FF  00  073F:0154
-s 150 15f ff
-s130 134 ff
073F:0130
073F:0131
073F:0132
073F:0133
073F:0134
-_
```

14) W Write

15) N Name

⇨ Steps used to save an assembly program:

1) Obtain length using "h"
2) Name your program using "n"
3) Put length on "CX"
4) Write your program on disk using "w"

```
C:\8086>debug
-a100
073F:0100 mov ah,02
073F:0102 mov dl,01
073F:0104 int 21
073F:0106 int 20
073F:0108
-h108 100
0208  0008
-n smile.com
-r cx
CX 0000
:0008
-w
Writing 00008 bytes
-
```

16) L Load

- First give name of file you want to load
- Load file using " l "

```
C:\8086>debug
-n smile.com
-l
-u 100 108
075A:0100 B402          MOV     AH,02
075A:0102 B201          MOV     DL,01
075A:0104 CD21          INT     21
075A:0106 CD20          INT     20
075A:0108 0000          ADD     [BX+SI],AL
-
```

Section #03 -- Git

1) Create GitHub account.
2) Create your Repository.
3) Download Git for Windows from: https://gitforwindows.org/
4) Install gitForWindows program.
5) Run Git Shell.

Git Commands:

✓ **Syntax**

```
git <options> command <options>
```

✓ **Configuration**

```
git config --global user.name 'Your name'
git config --global user.email your@domain.tld
```

✓ **Git Commands:**

Command	Description
git	To ensure that git is installed
git init	Create an empty Git repository or reinitialize an existing one
git clone	clone the repository into a new repo
git config	is a convenience function that is used to set Git configuration values on a global or local project level
git status	displays the state of the working directory and the staging area
git add	Change in the working directory to the staging area.
git add .	add all files to staging area
git rm	The git rm command can be used to remove individual files or a collection of files
git commit	Record changes to the repository

git show	Show various types of objects
git log	Show commit logs
git branch -l	list branch names
git branch <branch>	create new branch
git branch -r	remote branches
git branch -d <branch name>	to delete branch
git tag -l	list tag names
git tag -e <tag name> <sha for commit>	force edit of tag message
git checkout <branch name>	lets you navigate between the branches created by git branch.
git checkout -b <branch name>	create new branch and set head on it
git merge	to join two or more development histories together
git pull	The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content.
git push	Update remote refs along with associated objects