University of Central Punjab
(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)
FACULTY OF INFORMATION TECHNOLOGY

Computer Organization and Assembly Language

Lab 2	
Topic	1. Mov instruction 2. Viewing memory of dosbox 3. Declare variables

PART 1 (B)

Types of Registers:-

The registers are grouped into three categories:-

1. **General registers**
   - 1.1. *Data registers*
     - 1.1.1. ***AX*** is the primary accumulator.
     - 1.1.2. ***BX*** is known as the base register.
     - 1.1.3. ***CX*** is known as the count register.
     - 1.1.4. ***DX*** is known as the data register.
   - 1.2. *Pointer registers*
     - 1.2.1. Instruction Pointer ***IP***
     - 1.2.2. Stack Pointer ***SP***
     - 1.2.3. Base Pointer ***BP***
   - 1.3. *Index registers*
     - 1.3.1. Source Index ***SI***
     - 1.3.2. Destination Index ***DI***
2. **Control registers**
   - 2.1. Instruction Pointer and Flag register
3. **Segment registers**
   - 3.1. Code Segment ***CS***
   - 3.2. Data Segment ***DS***
   - 3.3. Stack Segment ***SS***
   - 3.4. Extra Segment ***ES***

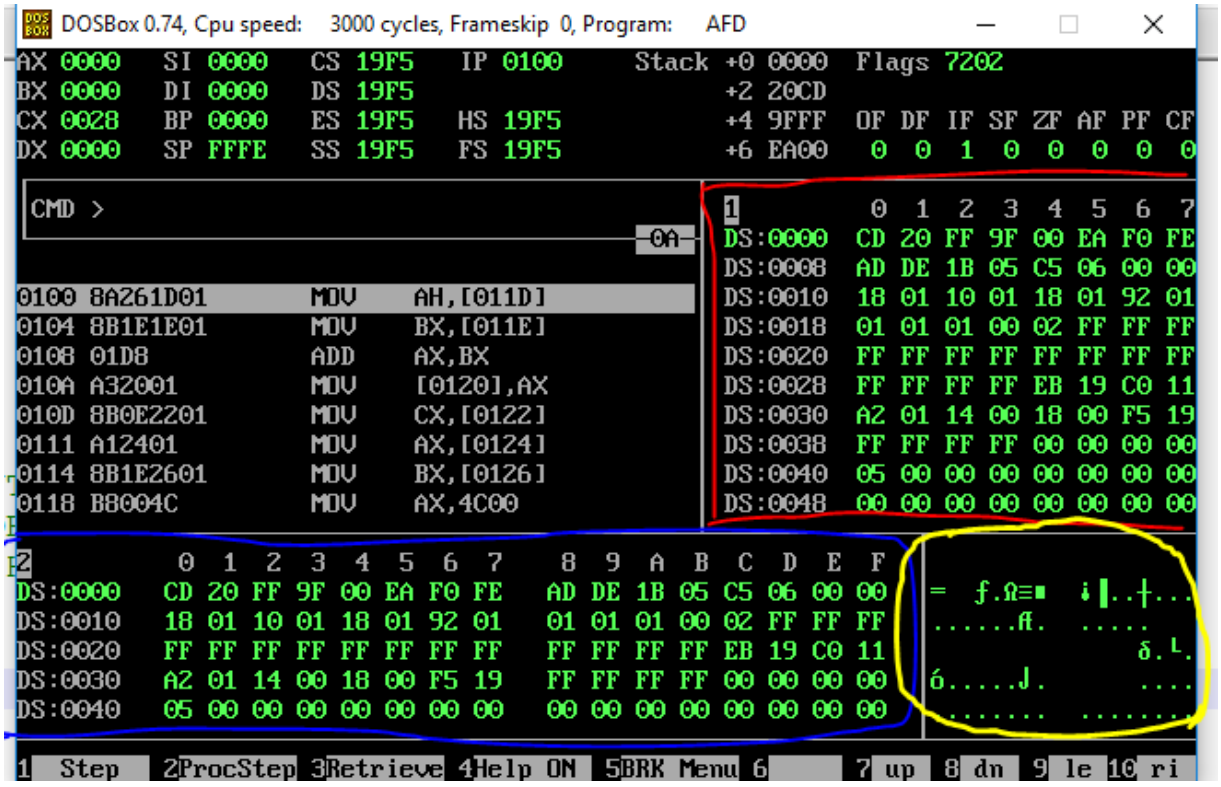
Types of variables

Type	No. of bits	Example declaration:
Byte	8	Num1: db 43
Word=> 2 bytes	16	Num2: dw 0xABFF
double word=> 2 words	32	Num3: dd 0xABCDEF56

Viewing memory in DOSBOX

Areas highlighted in red(memory 1) "m1" and blue (memory 2) "m2" are showing the memory contents. *Note:* Two copies of the same memory is displayed in the given window.

Area highlighted with yellow is showing the ascii values of the contents displayed in the memory m2.



Viewing sample variable in memory.

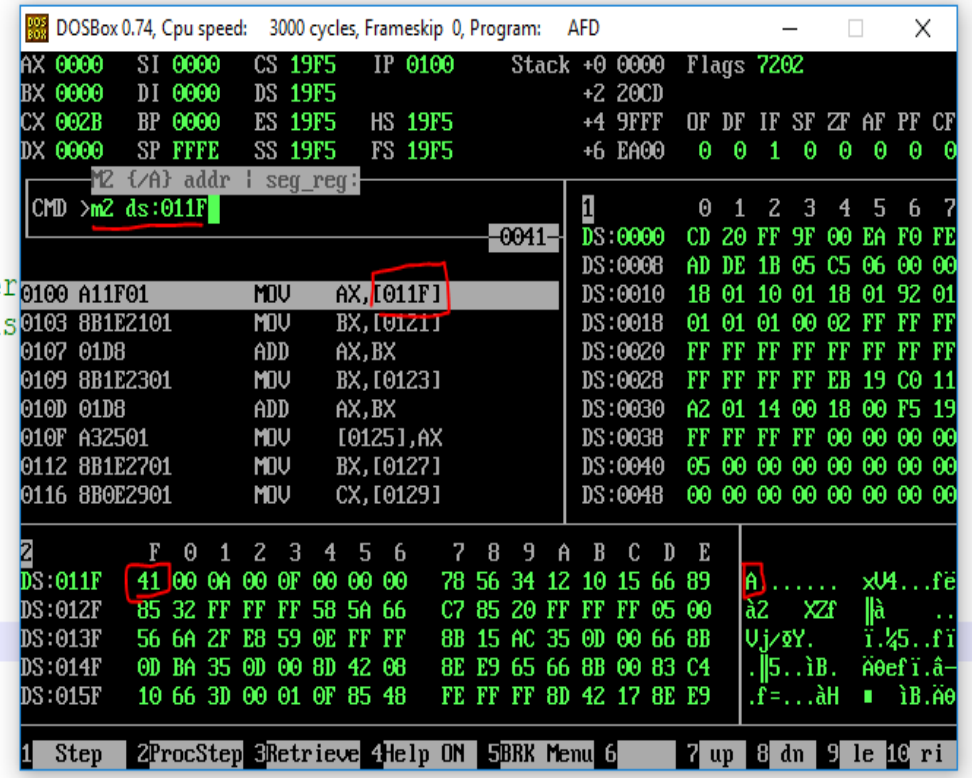
- To view memory from window m2 run the command "m2 ds:***Addressofvariable***" example: m2 ds:011F
- A variable with name "num1" is initialized with value 65 decimal.
  41 hex = 65 decimal is the ascii of "A".

```
[org 0x0100]

mov ax, [num1] ; load first number in ax
mov bx, [num2] ; load second number in bx
add ax, bx ; accumulate sum in ax
mov bx, [num3] ; load third number in bx
add ax, bx ; accumulate sum in ax
mov [num4], ax ; store sum in num4
mov bx,[num5];load lower 2 bytes of num5 in bx register
mov cx,[num5+2];load higher 2 bytes of num5 in cx regis

mov ax, 0x4c00 ; terminate program
int 0x21

num1: dw 65
num2: dw 10
num3: dw 15
num4: dw 0
num5: dd 0x12345678
```



Add Comment in code:

Use ; to start comment

```
[org 0x100]
mov al,[num1]
mov bh,[Num1]
add ax,bx

mov cl,[num2]
mov dx,[mynum] ;when using dw variables use a 16-bit register.

add cx,dx

mov ax,0x4c00
int 21h

num1: db 01100001b ;b is for binary
Num1: db 97 ;decimal by default, case sensitive names of variables
num2: db 0x61 ; 0x treats it as a hexadecimal number
mynum: dw 6100h; h at the end treats it as a hexadecimal number
temp: dw 0xABCD ;when using characters as a hex values, use 0x
```

or open your code in notepad++ select the code you want to comment/uncomment press ctrl+Q

Example 1:

```
[org 0x100]
mov al,9
mov bl,5
add al,bl
mov cl,-10
mov dl,11
add cl,dl
add al,cl

mov ax,0x4c00
int 21h
```

Example 2:

```
[org 0x100]
mov ax,75
mov bx,517
add al,bh
mov cl,200
mov dl,56
add cl,dl ;why we are getting 0 after addition ?

mov ax,0x4c00
int 21h
```

Example 3:

```
[org 0x100]
mov ax,75
mov bx,517
add al,bh
mov cl,200
mov dl,56
add cl,dl ;why we are getting 0 after addition ?

mov ax,0x4c00
int 21h
```

Example 4:

```
[org 0x100]
mov al,[num1]
mov bh,[Num1]
add ax,bx

mov cl,[num2]
mov dx,[mynum] ;when using dw variables use a 16-bit register.

add cx,dx

mov ax,0x4c00
int 21h

num1: db 01100001b ;b is for binary
Num1: db 97 ;decimal by default, case sensitive names of variables
num2: db 0x61 ; 0x treats it as a hexadecimal number
mynum: dw 6100h; h at the end treats it as a hexadecimal number
temp: dw 0xABCD ;when using characters as a hex values, use 0x

;uncomment the following line and then assemble again
; temp2: dw EFh ;why dosbox is showing error on this line???
```