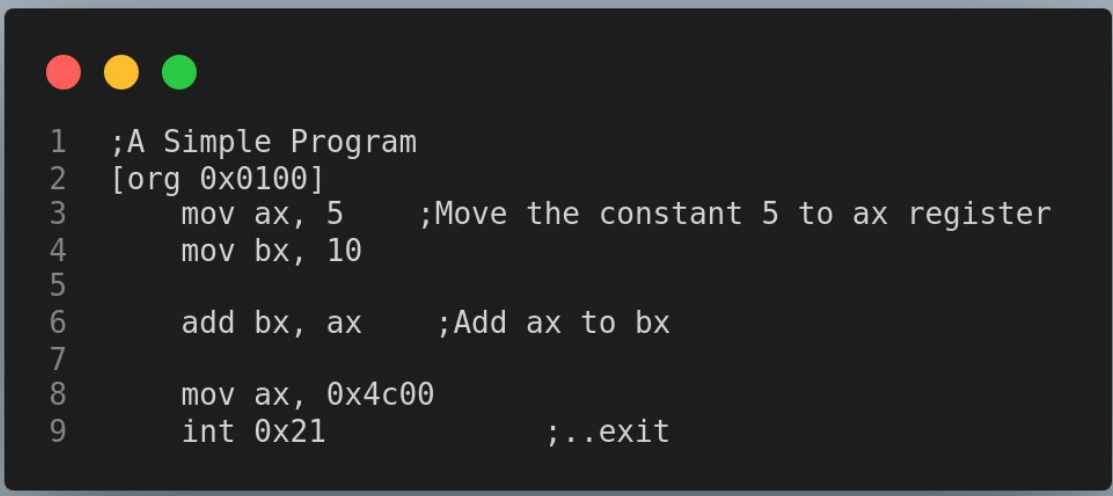


COAL_A_p200165_R3

Running Program In Dos Box:

In Lab 03 we have seen different registers and we have learned that how can we move data into RAM and also we have stored data In the RAM and we picked data from RAM using registers.

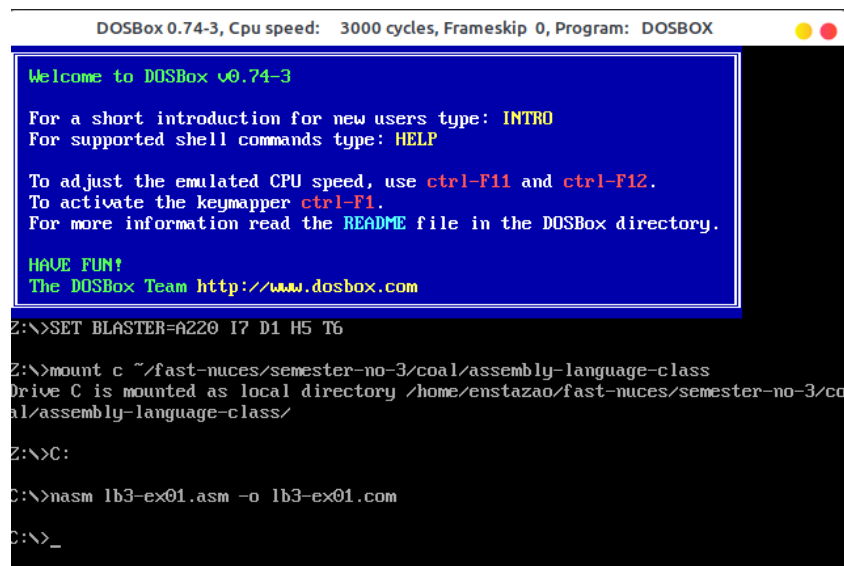
AX register that is also called an **accumulator register**. We used **ax** and **bx** that is **Base Register** we used them for these operations.

A screenshot of a DOSBox window with a dark background and light-colored text. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The text inside the window is assembly code for a simple program.

```
1 ;A Simple Program
2 [org 0x0100]
3     mov ax, 5      ;Move the constant 5 to ax register
4     mov bx, 10
5
6     add bx, ax      ;Add ax to bx
7
8     mov ax, 0x4c00
9     int 0x21        ;..exit
```

This is the first program we have executed. We have executed this program in **DOSBOX** using **nasm** assembler.

If there is no error in the code then code will be assembled successfully.

A screenshot of the DOSBox interface. The top bar shows 'DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX'. The main window has a blue title bar and a black background with white text. A white-bordered box contains the welcome message. Below it, the command prompt shows the execution of 'nasm' to assemble the program.

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

Welcome to DOSBox v0.74-3

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount c ~/fast-nuces/semester-no-3/coal/assembly-language-class
Drive C is mounted as local directory /home/enstazao/fast-nuces/semester-no-3/coal/assembly-language-class/

Z:\>C:

C:\>nasm lb3-ex01.asm -o lb3-ex01.com

C:\>_
```

Second Program:

```
1 ; A Program to swap two variables
2 [org 0x01000]
3     mov ax, 5    ;Move the value 5 to ax register
4     mov bx, 10   ;Move the value 10 to ax register
5
6     mov cx, ax   ;Move the value of ax to cx register
7     mov ax, bx   ;Move the value of bx to ax register
8     mov bx, cx   ;Move the value of cx to ax register
9
10    mov ax, 0x4c00 ;Move the 4c00 to ax register
11    int 0x21      ;..exit
```

We have also used **AFD** which is an **Advance free debugger** that is used to debug your code.

This is the **AFD Debugger** that is used to debug the code. Also we have seen listing file in the lab that contain **assembled machine code**.

The last thing we have done in LAB03 is the Lab Task. In the Lab task we have to calculate the square of 6.

LAB TASK:

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount c ~/fast-nuces/semester-no-3/coal/assembly-language-class
Drive C is mounted as local directory /home/enstazao/fast-nuces/semester-no-3/coal/assembly-language-class/

Z:\>nasm le01.asm -o le01.com
Illegal command: nasm.

Z:\>C:

C:\>nasm le01.asm -o le01.com

C:\>afd le01.com

AFD-Pro is done

C:\>nasm ltask.asm -o ltask.com -l ltask.lst

C:\>afd ltask.com

AFD-Pro is done

C:\>I_
```


REG	VALUE	REG	VALUE	REG	VALUE	REG	VALUE
AX	0000	SP	FFFE	SS	19F5	FS	19F5

CMD	>
0100	B80500 MOV AX,0005
0103	BB0A00 MOV BX,000A
0106	01D8 ADD AX,BX
0108	B804C MOV AX,4C00
010B	CD21 INT 21
010D	C3 RET
010E	89D0 MOV AX,DX
0110	89DA MOV DX,BX

DS:0000	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DS:0000	CD	20	FF	9F	00	EA	F0	FE	AD	DE	1B	05	C5	06	00	00
DS:0008	AD	DE	1B	05	C5	06	00	00	01	01	01	00	02	FF	FF	FF
DS:0010	1B	01	10	01	18	01	00	00	FF	FF	FF	FF	FF	FF	FF	FF
DS:0018	01	01	01	00	02	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
DS:0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
DS:0028	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
DS:0030	A2	01	14	00	18	00	F5	19	FF	FF	FF	FF	FF	FF	FF	FF
DS:0038	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
DS:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
DS:0048	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

DS:0000	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DS:0000	CD	20	FF	9F	00	EA	F0	FE	AD	DE	1B	05	C5	06	00	00
DS:0010	1B	01	10	01	18	01	92	01	01	01	01	00	02	FF	FF	FF
DS:0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	EB	19	C0	11
DS:0030	A2	01	14	00	18	00	F5	19	FF	FF	FF	FF	FF	FF	FF	FF
DS:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 1



```
1 ;A program to calculate the sqaure of Six
2 [org 0x0100]
3
4     mov ax, 0
5     mov cx, 6
6
7     outerloop:
8         add ax, [num1]
9
10        sub cx, 1
11        jnz outerloop
12
13        mov ax, 0x4c00
14        int 0x21
15 num1: dw 6
```

I have used loop in this task. Basically I used labels to create repetition. **Jnz** flag will set if the previous arithmetic or logical operation produce result of zero. If the zero flag is set the **after jnz** instruction repetition is created else program will be executed and no jump is made. That's all we have done in our lab three.

Name : Jawad Ahmed

Roll No: 20P-0165

Section : 3A