

National University of Computer and Emerging Sciences

Lab Manual

Computer Organization and Assembly Language



Lab 01

Instructor	Hazoor Ahmad
Class	CS3
Sections	A1, D1, H1, K2
Semester	Fall 2022

Fast School of Computing

FAST-NU, Lahore, Pakistan

Objectives

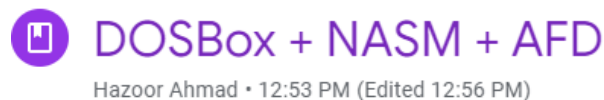
- To setup DOSBOX, NASM and AFD
- Running your first program in Assembly
- Practicing few basic programs in Assembly
- Explaining basic functions in debugger

Contents

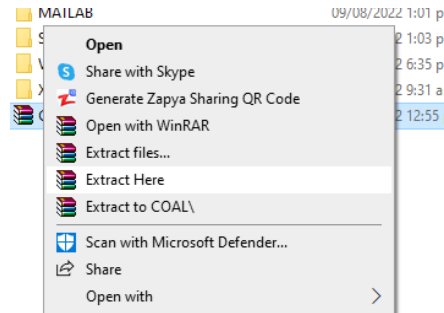
Objectives	2
ACTIVITY 1: INSTALLATION & IMPORTANT COMMANDS	2
ACTIVITY 2: RUNNING YOUR FIRST PROGRAM	5
ACTIVITY 3 PRACTICE QUESTION	8
ACTIVITY 4: PRACTICE QUESTION	8
APPENDIX:	8
REFERENCES	9

ACTIVITY 1: INSTALLATION & IMPORTANT COMMANDS

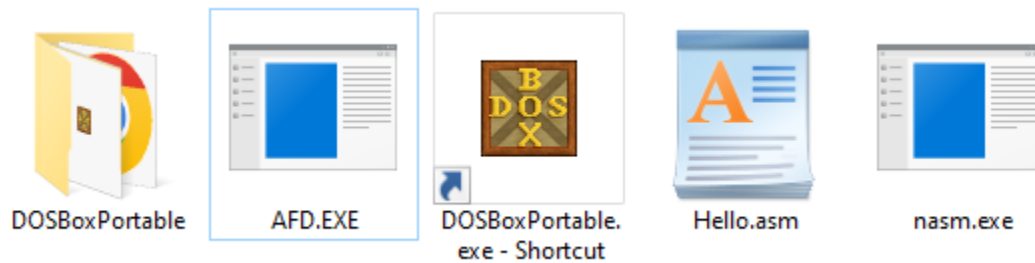
1. First, make sure that there is no folder named “COAL” in “D:\” Drive of your system.
2. Once you ensure step 1, Sign into the classroom for downloading the set of software required for this lab. Your Class Post seems like below



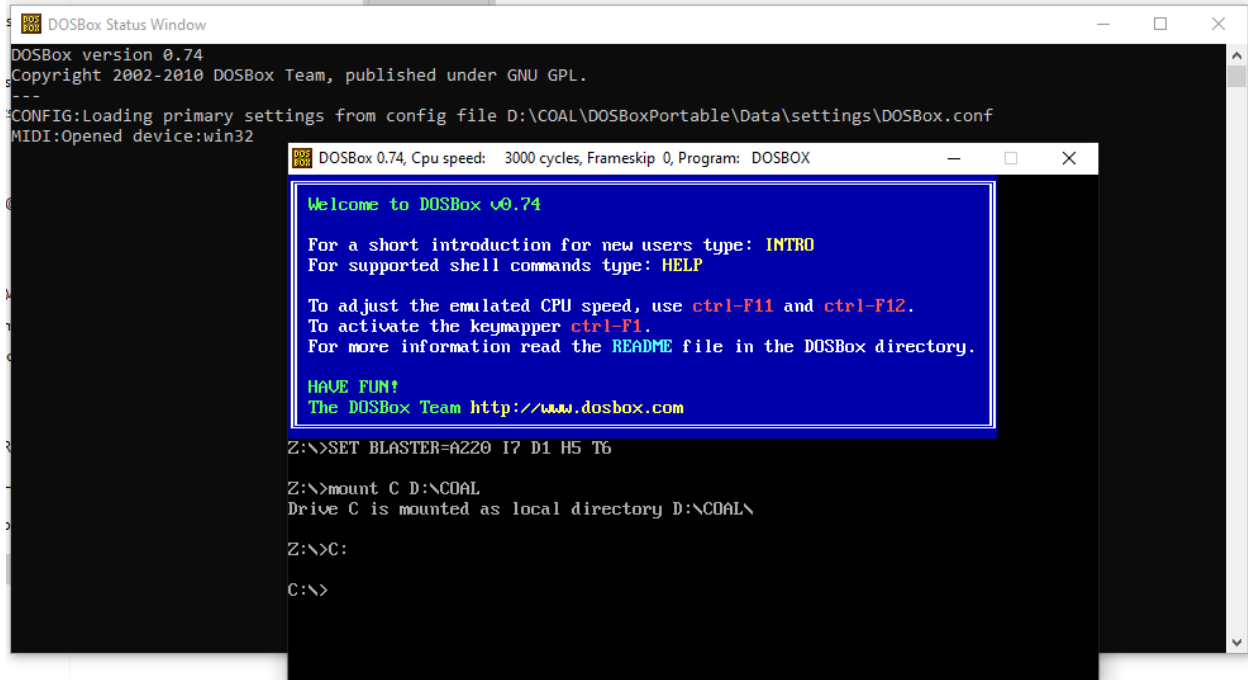
3. Download and take the file to the “D:\” Drive of your system.
4. Right Click on the file and press “Extract Here” [if you have WinRAR installed] as shown.



5. After extracting you will see a “D:\COAL” folder in “D:” Drive of your system.
6. Open that folder and you will see the following contents



7. This folder has everything ready for you, you just click on “DOSBoxPortable.exe” to start.
8. Once DOSBox is started it will look like the following:



9. Your environment [already set] has following abstractions:
 - a. DOSBox is already mounted at “D:\COAL”.
 - b. You must make all your assembly language programs in the folder “D:\COAL”.

- c. In case you want to change the path see “MOUNT Command” in step 10.
- d. DOSBox requires “nasm.exe” and “afd.exe” which should be present to the folder you mounted.

10. **MOUNT Command:** The format of MOUNT command is as follows:

```
mount<space>Mount_Name<space>Mount_Path
```

you can specify any name to the mount command and any existing path from your system. One example is shown below:

```
mount<space>C<space>D:\COAL
```

You can specify more than one mount points to the DOSBox but keep “nasm.exe” and “afd.exe” at every mount point.

You can change mount point by specifying a Mount_Name a suffix of “:”, as shown below

```
C:
```

11. **NASM Command:** The Netwide Assembler, NASM, is an 80x86 and x86-64 assembler designed for portability and modularity. You can perform following operations using assembler

Compile: To compile “hello.asm” program type the following command

```
nasm<space>hello.asm
```

Assemble: To assemble “hello.asm” program type the following command

```
nasm<space>hello.asm<space>-o<space>hello.com
```

Listing File: To generate listing file from “hello.asm” program type the following command

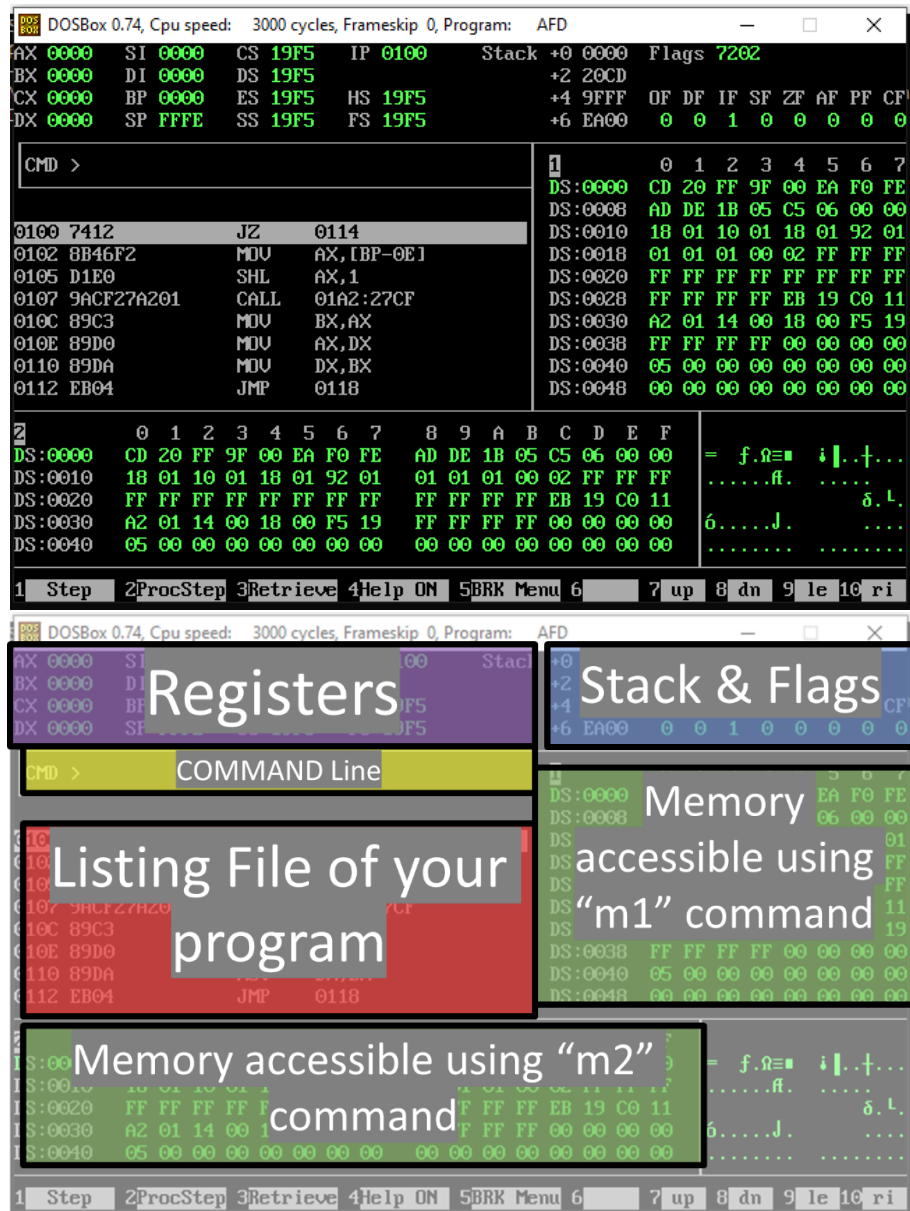
```
nasm<space>hello.asm<space>-l<space>hello.lst
```

Other Controls: To have more controls type the following command

```
nasm<space>-h
```

16. **AFD Command:** To launch the debugger of “hello.com” file generated in your environment you should run the following command

```
afd<space>hello.com
```



- To scroll into the listing file press "F1"
- To see specific memory location from M1 or M2 memory just type m1 or m2 followed by address of that specific memory location.
- To quit from AFD type the following "quit" in COMMAND Line of AFD.

ACTIVITY 2: RUNNING YOUR FIRST PROGRAM

Follow these steps to run your first program:

1- Copy/paste following code in notepad

```
; this is a comment. Comment starts with semicolon
; this program adds three numbers in registers
```

```
[org 0x0100] ;we will see org directive later
```

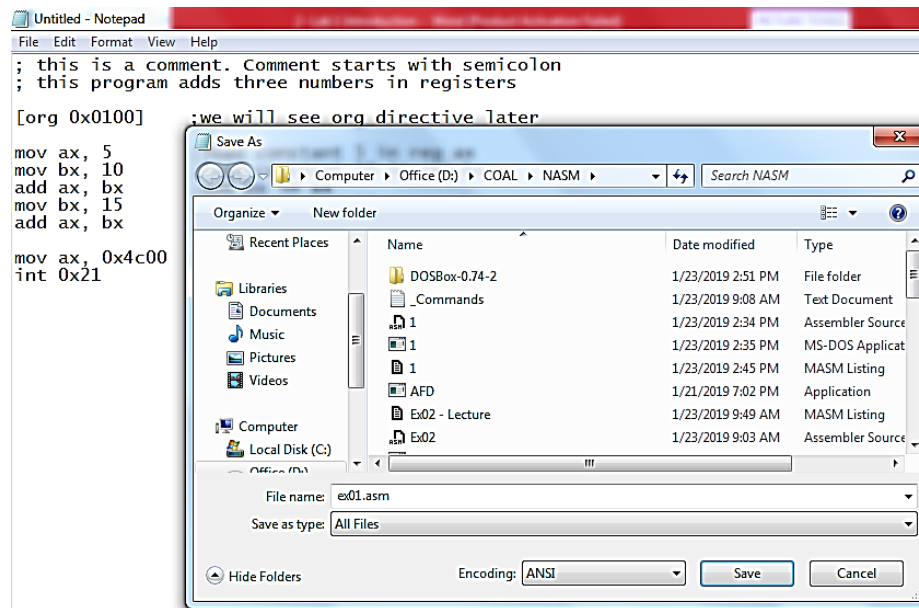
```

mov ax, 5      ; AX = 5
mov bx, 10     ; BX = 10
add ax, bx     ; AX = AX + BX
mov bx, 15     ; BX = 15
add ax, bx     ; AX = AX + BX

mov ax, 0x4c00 ;terminate the program
int 0x21

```

2- Save this file as “ex01.asm” in your folder e.g. “D:\COAL”:



nasm ex01.asm -o ex01.com -l ex01.lst

- 3- Above command will assemble your code and create ex01.com and ex01.lst files. Open ex01.lst file in notepad.
- What is opcode of instruction “mov ax, someConstant”
 - Verify the above opcode everywhere the instruction has been used.
 - What does “B80500” mean?
 - Verify the opcode of instruction “mov bx, someConstant” throughout the machine code.
 - What is the offset of first instruction?
 - Why are offsets of second and third instructions 3 and 6?
 - What should be the size of ex01.com file?
 - Right click ex01.com and verify its size.
- 4- Open DOSBox (by double clicking dosbox.exe), following window will appear:

```

DOSBox 0.74-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

Welcome to DOSBox v0.74-2

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 D://COAL//NASM
Drive C is mounted as local directory D://COAL//NASM

Z:\>C:

C:\>

```

- 5- Write following command and press enter.

Afd ex01.com

(Your AFD.exe should be in same directory where we have installed everything)

```

DOSBox 0.74-2, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX 0000 SI 0000 CS 19F5 IP 0100 Stack +0 0000 Flags 7202
BX 0000 DI 0000 DS 19F5      +2 20CD
CX 0012 BP 0000 ES 19F5 HS 19F5  +4 9FFF OF DF IF SF ZF AF PF CF
DX 0000 SP FFEE SS 19F5 FS 19F5  +6 EA00  0 0 1 0 0 0 0 0

CMD >

0100 B80500 MOV AX,0005
0103 BB0A00 MOV BX,000A
0106 01D8 ADD AX,BX
0108 BB0F00 MOV BX,000F
010B 01D8 ADD AX,BX
010D B804C MOV AX,4C00
0110 CD21 INT 21
0112 0000 ADD [BX+SI],AL

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 18 01 10 01 18 01 92 01 01 01 01 00 02 FF FF FF
DS:0018 01 01 01 00 02 FF FF FF DS:0020 FF FF FF FF FF FF FF FF
DS:0028 FF FF FF FF EB 19 C0 11 DS:0030 A2 01 14 00 18 00 F5 19
DS:0038 FF FF FF FF 00 00 00 00 DS:0040 05 00 00 00 00 00 00 00
DS:0048 00 00 00 00 00 00 00 00

2 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 = f.Ω≡ i|.+.
DS:0010 18 01 10 01 18 01 92 01 01 01 01 00 02 FF FF FF .....f. ....
DS:0020 FF FF FF FF FF FF FF FF FF FF FF FF EB 19 C0 11 .....δ.L.
DS:0030 A2 01 14 00 18 00 F5 19 FF FF FF FF 00 00 00 00 ó.....J. ....
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 10 ri

```

- 6- Above command will open the debugger and load your ex01.com file in it.
- What is the value of IP register? And what will be its effect?
 - Note the initial values of data registers
 - Press F1 and watch the values of data registers

ACTIVITY 3 PRACTICE QUESTION

Modify this program to generate the sum of first five entries of table of 3, using registers, and watch its execution in the debugger.

Help: [Approach 1] Can you do this using two registers only? [Approach 2] Can you do this using one register only if we have **add ax, 3** available in our instruction set? Try both of these approaches and watch the first five entries of table of 3 in AX.

ACTIVITY 4: PRACTICE QUESTION

Write a program that rotates the value of three registers clockwise twice i.e. given these initial values: ax=10, bx=20, cx=30.

Help: First draw data registers on paper. Solve this problem on paper then write your code accordingly. After writing your code verify the execution on paper first then check the execution on AFD. You may use DX.




APPENDIX:

Listing File:

Listing file is created by nasm when you enter “-l list_file_name.lst” during assembling the program, such as, if you enter the following:

nasm code.asm -o output.com -l list_file.lst

The above command will now generate a listing file named “list_file.lst” along with the com file. Listing File will look something like this:

1		
2		[ORG 0x0100]
3		
4		
5	00000000 B80100	mov ax, 1
6	00000003 89C3	mov bx, ax
7	00000005 81C30300	add bx, 3
8		
9	00000009 B8004C	mov ax, 4c00h
10	0000000C CD21	int 0x21
		
	Instruction offset	This is our assembly source code.
		
	This is our actual machine code of each instruction, Here in listing file the machine code of each instruction is being shown in hexadecimal	

Difference between Mnemonic and Opcode:

MNEMONIC: Human Readable words. The assembly keywords, such as, mov, add, sub, etc are MNEMONICS for programmers because they are easily understood by them. We use these MENMONICS to write programs because we can easily remember mnemonics. Mnemonics cannot be executed by the CPU, so mnemonics are always converted into some opcode which can be executed by the CPU.

OPCODE: It is a number interpreted by the CPU that represents the operation to perform. For example in the above listing file, the opcode for moving an immediate operand into AX register is B8. Can you identify the opcode for adding an immediate value into BX from the above listing file?

Purpose of [ORG 0x0100]:

It simply tells the nasm that the instructions of our program should placed at the start 256th byte of code segment. (The first 256 bytes are to be skipped). Note that 0x0100 is a hexadecimal number whose decimal value is 256. The reason we skip the first 256 bytes is because these bytes have some important piece of code that we do not want to overwrite with our own. This will be further clarified in some later lab session.

REFERENCES

- ["http://www.dosbox.com/download.php?main=1](http://www.dosbox.com/download.php?main=1)
- <http://sourceforge.net/projects/nasm>
- <http://www.nasm.us/>
- [http://www.programmersheaven.com/download/21643/download.aspx \(AFD\)](http://www.programmersheaven.com/download/21643/download.aspx (AFD))