**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

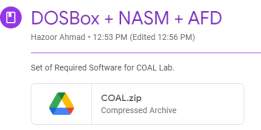
• Practicing few basic programs in Assembly

• Explaining basic functions in debugger

Contents

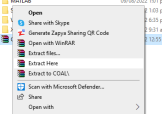
Objectives .....................................................................................................................................................2 ACTIVITY 1: INSTALLATION & IMPORTANT COMMANDS .........................................................................2 ACTIVITY 2: RUNNING YOUR FIRST PROGRAM.........................................................................................5 ACTIVITY 3 PRACTICE QUESTION ..............................................................................................................8 ACTIVITY 4: PRACTICE QUESTION .............................................................................................................8 APPENDIX:.................................................................................................................................................9 REFERENCES............................................................................................................................................10

**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

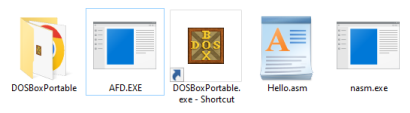


3. Download and take the file to the “D:\” Drive of your system.

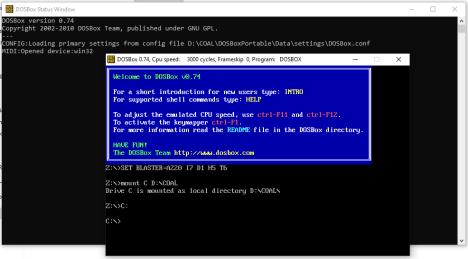
4. Right Click on the file and press “Extract Here” [if you have WinRAR installed] as shown.



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

12. nasm<space>hello.asm

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

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

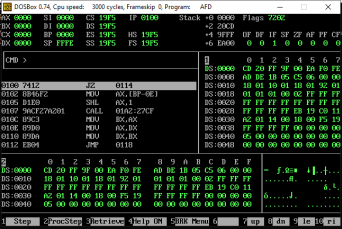
**Listing File**: To generate listing file from “hello.asm” program type the following command 14. nasm<space>hello.asm<space>-l<space>hello.lst

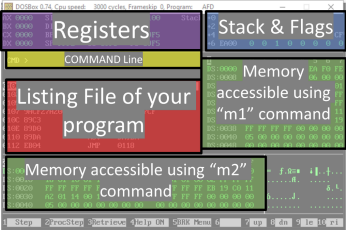
**Other Controls**: To have more controls type the following command

15. nasm<space>-h

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

17. 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.

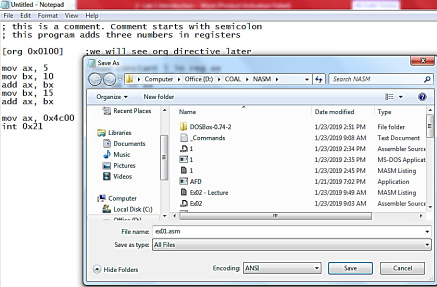
• To quit from AFD type the following “quit” in COMMAND Line of AFD.

**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 NASM folder e.g. “D:\COAL\NASM”: 

3- Go to NASM installation directory ( e.g. “D:\COAL\NASM”). Double click **nasmpath.bat** (batch file) and type following command there. (Your .asm file and nasm should be in one folder)

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

4- Above command will assemble your code and create ex01.com and ex01.lst files. Open ex01.lst file in notepad.

a. What is opcode of instruction “mov ax, someConstant”

b. Verify the above opcode everywhere the instruction has been used. c. What does “B80500” mean?

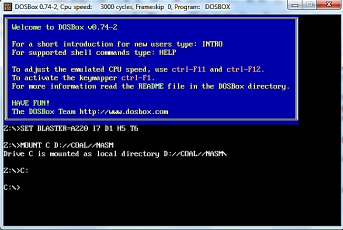
d. Verify the opcode of instruction “mov bx, someConstant” throughout the machine code.

e. What is the offset of first instruction?

f. Why are offsets of second and third instructions 3 and 6?

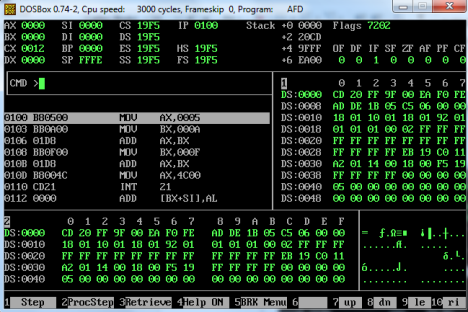
g. What should be the size of ex01.com file?

h. Right click ex01.com and verify its size.

5- Open DOSBox (by double clicking dosbox.exe), following window will appear: 6- Write following command and press enter.

**Afd ex01.com**

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



7- Above command will open the debugger and load your ex01.com file in it. a. What is the value of IP register? And what will be its effect?

b. Note the initial values of data registers

c. 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.

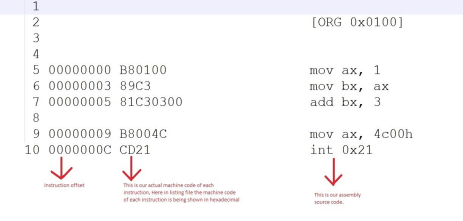
**APPENDIX:**

**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:



**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://sourceforge.net/projects/nasm

• http://www.nasm.us/

• http://www.programmersheaven.com/download/21643/download.aspx (AFD)