National University of Computer and Emerging Sciences



Laboratory Manual

for

Computer Organization and Assembly Language Programming

(EL 213)

|  |  |
| --- | --- |
| Course Instructor | Ms. Farwa Batool |
| Lab Instructor(s) | Ms. Ayesha Ajaz  Ms. Hamna Waseem |
| Section | CS-D |
| Semester | Fall 2019 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

## Objectives

After performing this lab, students shall be able to:

* Learn about the tools required
* Be familiarized with the coding environment
* Have basic idea of assembly language programming

## ASSEMBLY TOOLS AND ENVIROMENT

There are 3 tools that you’ll need for this course:

Nasm (An assembler)

Afd (A debugger)

Dosbox (A virtual box)

Follow given steps:

1. You can get these tools from

\\cactus\Xeon\Spring2019\Assembly Labs\Assembly\_Tools

Create a folder on your C drive named as “**Assembly**”. Now copy DosboxPortable , Nasm and Afd from the Assembly Tools folder and paste in newly created folder.

2. Open DosBoxPortable folder and run executable file named “DOSBoxPortable.exe”.

3. By default the drive is Z:\ . Type command **mount X C:\Assembly**. This will virtually create a drive X and C:\Assembly will contain contents for this drive.

4. Still drive is Z:\, X is only mounted, to change drive or directory write **X:** and then press enter. You are done.

5. You can write these mount commands at the end of .conf file in portable dropbox folder having the path “DosboxPortable\Data\settings\dosbox.conf”, it will save your time. Next time when you open dropbox, you don’t have to mount again and again.

6. Once you have mounted your drive, make sure that you have nasm and afd in the same mounted drive path. Now create an assembly file with the extension “<file\_name>.asm” in the same mounted drive where both nasm and afd are placed.

7. To assembly your file, firstly you have to mount your drive to the path which contains three files named nasm, afd and “\*.asm”e.g.,

**Mount x “C:\Assembly”**

**X:**

You can write the above two commands in “Dosbox.conf” file so that you don’t have to mount the drive again when you open DropBox. They will already be executed when DOSBox starts everytime

8. Once you have mounted your drive you can assemble your program by writing the following command:

**Nasm \*.asm –o \*.com –l \*.lst** //Here \* means your file name

9. This will create one com file and one listing file if there are no errors. You can now debug your program by opening the \*.com file in afd debugger using the following command:

**Afd \*.com**

10. This will open afd debugger. You can step through your code and see the value of your registers changing as you step through the written code line by line. You can use F1 or F2 to step through your code line by line.

**Exercise 1:** Write an assembly program that will sum first 10 prime numbers using only two registers ax and bx. You have to use ax to accumulate the sum whereas bx will be used to get the next hardcoded number in the list of first 10 Prime numbers. Store the final result (sum of 10 Prime numbers) in dx register.

Prime Numbers: 2,3,5,7,11,13,17,19,23,29 (Sum=129)

**Exercise 2:** If ax is initialized with 0x1212 and bx with 0x2000, Find the number ‘X’ that together with ax, makes sum equal to bx.

i.e. ax+X=bx. Find the value of X.

Once you’ve found the value of X, move it to cx.

**Exercise 3:** See the values of carry flag, zero flag and sign flag after executing these instructions one by one:

1. ax=20, bx=0xffff add ax, bx
2. ax=20, bx=0x20 sub ax, bx
3. ax=20 sub ax, ax

## Good to Read:

### 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.

**Purpose of [ORG 0x0100]:** It simply tells the nasm that the instructions of our program should place 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.