CSE 331L / EEE 332L: Microprocessor Interfacing & Embedded System

Section: 7, Fall 2019

Instructor's email: tasmia.shahidi@northsouth.edu

You will be introduced to assembly language programming and to the emu8086 emulator software as editor and assembler for your assembly language programming.

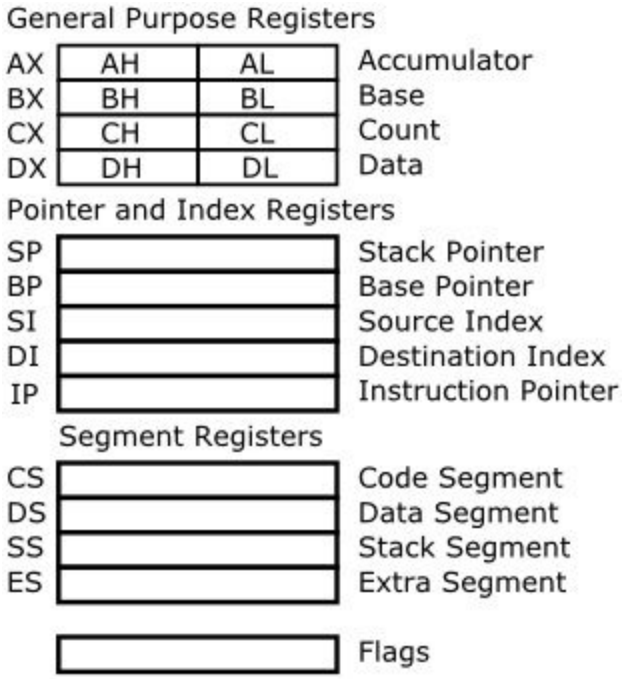
Steps required to run an assembly program:

1. Write the necessary assembly source code
2. Save the assembly source code
3. Compile/Assemble source code to create machine code
4. Emulate/Run the machine code

What is Assembly Language:

Processors only understand machine language instructions which are strings of 0's and 1's. Assembly language is the low-level language that uses symbolic names to represent operations, registers, memory location etc. and can be converted to machine language instructions with the help of Assembler (a program that converts each assembly language statement into a single machine language instruction).

Registers of MPU 8086:



Total 14 registers each with 16 bits size.

- AX - The Accumulator register (divided into AH / AL). Data read/write operations done with AX(16 bits) or AL(8 bits)
- BX - The Base Address register (divided into BH / BL).
- CX - The Counter register (divided into CH / CL).
- DX - The Data register (divided into DH / DL). It holds the results during arithmetic operation. Holds address of variable port during I/O operations.

4 general purpose registers (AX, BX, CX, DX) are made of two separate 8 bit registers, for example if AX= 0011000000111001b, then AH=00110000b (higher order byte) and AL=00111001b (lower order byte). Therefore, when you modify any of the 8 bit registers 16 bit register is also updated, and vice-versa. The same is for other 3 registers, "H" is for high and "L" is for low part.

Since registers are located inside the CPU, they are much faster than memory. Accessing a memory location requires the use of a system bus, so it takes much longer. Accessing data in a register usually takes no time. Therefore, you should try to keep variables in registers. Register sets are very small and most registers have special purposes which limit their use as variables, but they are still an excellent place to store temporary data of calculations.

Writing of your first Assembly Code:

This class will introduce two instructions and will serve as the basis for your first assembly program. The following table shows the instruction name, the syntax of its use, and its description. The operands heading refers to the type of operands that can be used with the instruction along with their proper order.

- REG: Any valid register
- Memory: Referring to a memory location in RAM
- Immediate: Using direct values.

Instruction	Operands	Description
MOV	REG, memory memory, REG REG, REG memory, immediate REG, immediate	The MOV instruction cannot: ● set the value of the CS and IP registers. ● copy value of one segment register to another segment register (should copy to general register first). ● copy immediate value to segment register (should copy to general register first). Algorithm: operand1 = operand2
ADD	REG, memory memory, REG REG, REG memory, immediate REG, immediate	Algorithm: Operand1 = operand1 + operand2

TASK 1

Write the following code in emu8086 editor:

```
ORG 100H

MOV AX,2

MOV BX,2

ADD AX,BX

MOV CX,AX

RET
```

The first line of this program, ORG 100H (origin), is a necessary requirement for all assembly programs written in emu8086. It tells the assembler what address it will be loaded at.

";" is used for comments.

Your program should also always end with the RET (return) instruction. This instruction basically gives back control of CPU and system resources back to the operating system. The RET statement will be used in further classes.

This program basically adds two numbers stored in two separate registers. The final result is stored in a third register. Assemble this program and run it. Follow the in class lecture regarding the use of the emulator and its various features and debugging techniques.

Home Task:

- ➢ Download and install emu8086
- ➢ Experiment with the instructions you learnt