**East West University**

**Assignment**

Course Code & Title: CSE442 Microprocessor & Microcontroller

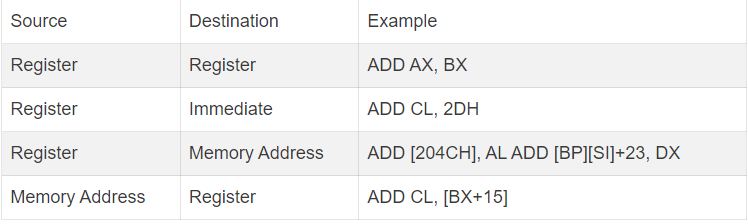
Section: 02

Subject: LAB 1& 2

Student Name: Abdullah al nayeem

Student ID: 2018-1-60-086

**ADD:** This instruction adds the data of destination and source operand and stores the result in destination. Both operands should be of same type i.e. words or bytes otherwise assembler will generate an error. It supports following operands:



Example:

**ORG 100h**

**.MODEL SMALL**

**.CODE**

**MOV AL, 10H ;Sets AL to 10H**

**MOV BH, 75H ;Sets BX to 23H**

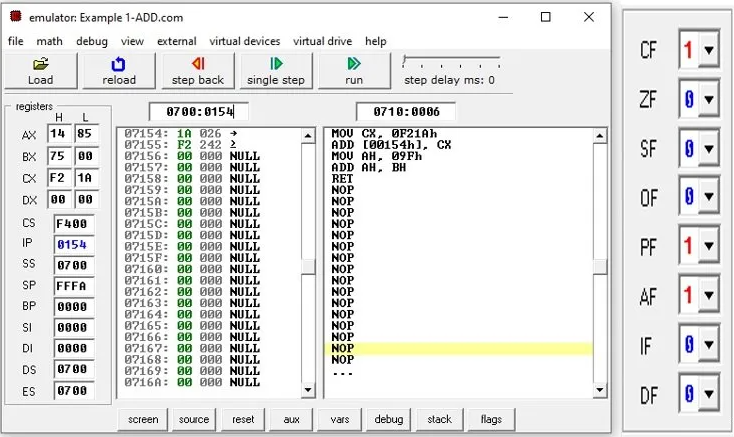
**ADD AL, BH ;Store the sum of AL and BX in AL**

**MOV CX, 0F21Ah ;Set CX to 0F21Ah**

**ADD [0154H], CX ;Store sum of CX data and data at memory address DS:0154 into the same memory address**

**MOV AH, 9FH ;Sets AH to 9FH**

**ADD AH,BH ;Store sum of AH and BH into BH**



**ADC:** The ADC and ADD instruction perform the same operation of addition. The only difference is that ADC instruction also adds the carry flag bit to the sum of two operands.

Example:

**ORG 100h**

**.MODEL SMALL**

**.CODE MOV AL, 7DH ;Sets AL to 7DH**

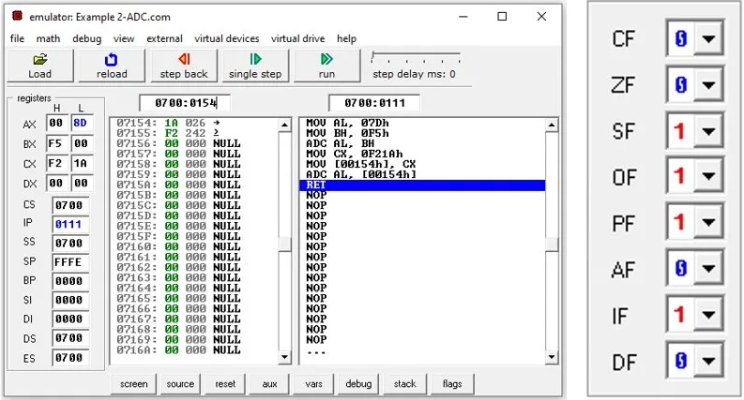
**MOV BH, 0F5H ;Sets BH to F5H**

**ADC AL, BH ;Store the sum of AL and BH in AL**

**MOV CX, 0F21Ah ;Set CX to 0F21Ah**

**MOV [0154H], CX ;CX=CX+ DS:0154H**

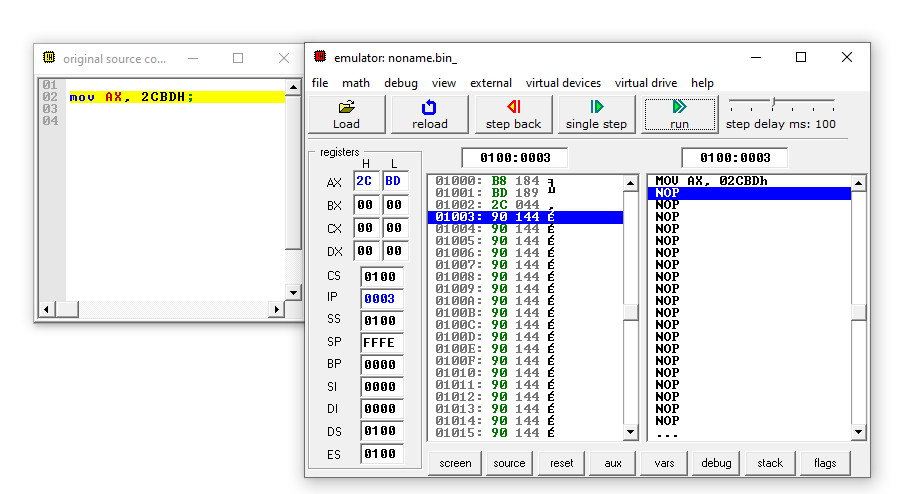
**ADC AL, [0154H] ;AL=AL+ DS:0154H**



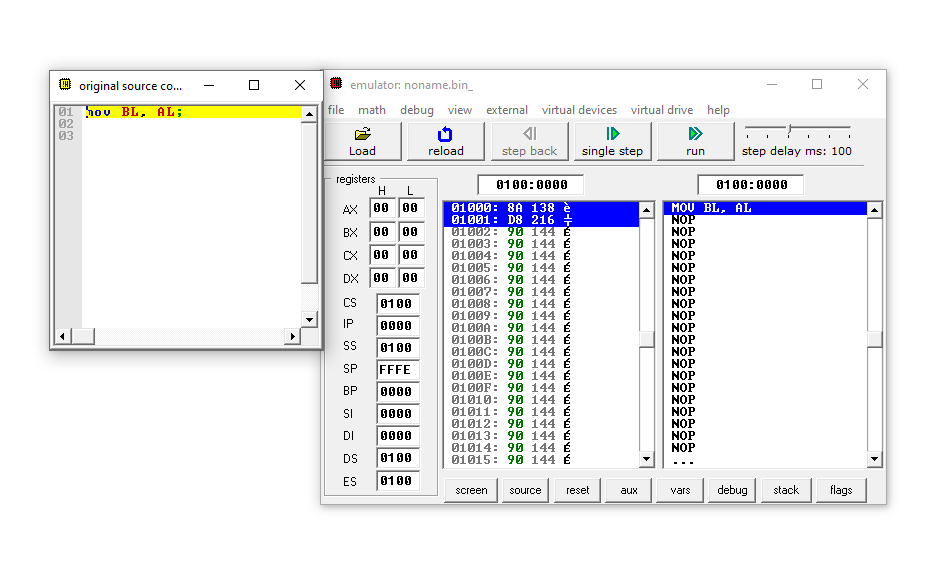
**MOV:** This “MOV” instruction is used to copy the byte data or word from the source to the destination. (The structure of the instruction is **mov destination, source;**)The provided number divided into two segments. The higher number goes into the AH and lower number goes to the AL segment. **mov AX, 2CBDH;** Here 2CBD divided into 2C BD and AH = 2C AL= BD. Again, **mov BL, AL;** here the value of AL will be copied into the BL register. So, BL = BD

Example:

**mov AX, 2CBDH;**



**mov BL, AL;**



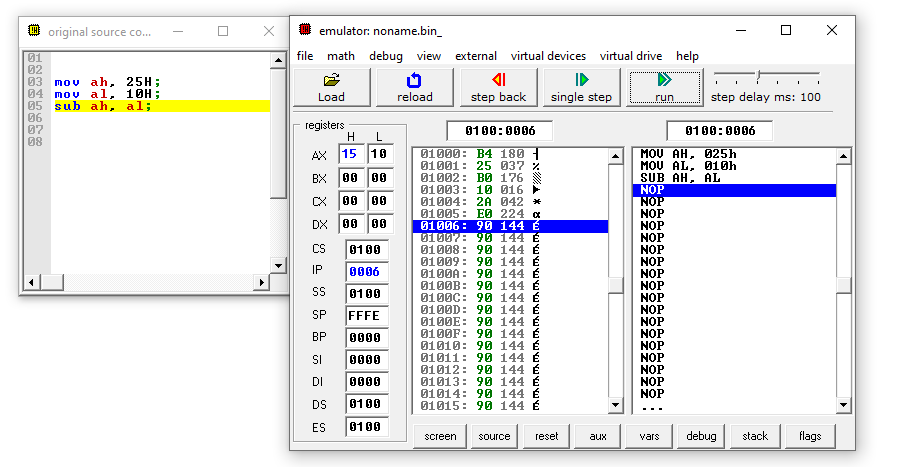
**SUB:** This instruction “SUB” is used to subtract the data of two registers and put the value into a destination register.

Example:

**mov ah, 25H;**

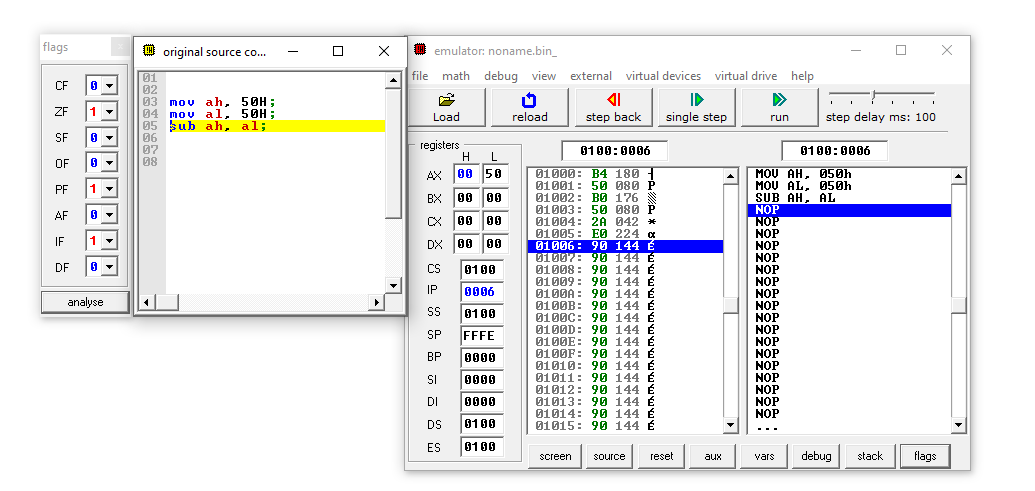
**mov al, 10H;**

**sub ah, al;**

**mov ah, 50H;**

**mov al, 50H;**

**sub ah, al;**



**INC:** This instruction “INC” is used to manipulate the data. This instruction increments the value by 1. Suppose, we have **mov DX, 2B9CH;** And after the mov instruction is executed DH = 2B and DL = 9C. After we execute **inc DX** instruction then the lower level number will be DL = 9D.

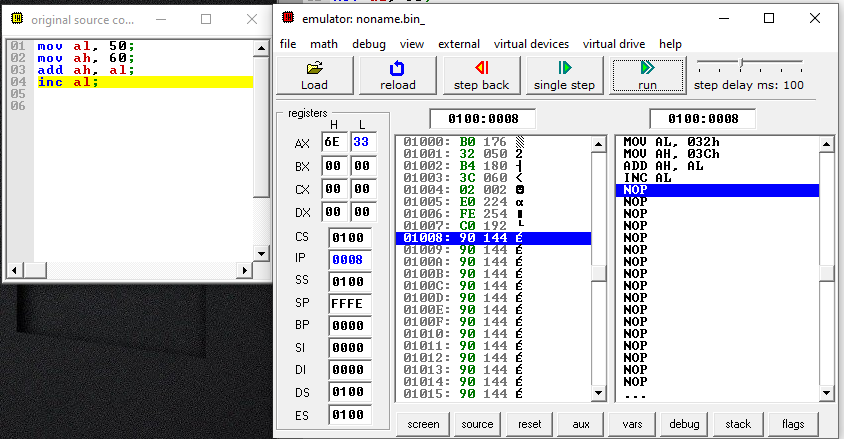
Example:

**mov al, 50;**

**mov ah, 60;**

**add ah, al;**

**inc al;**

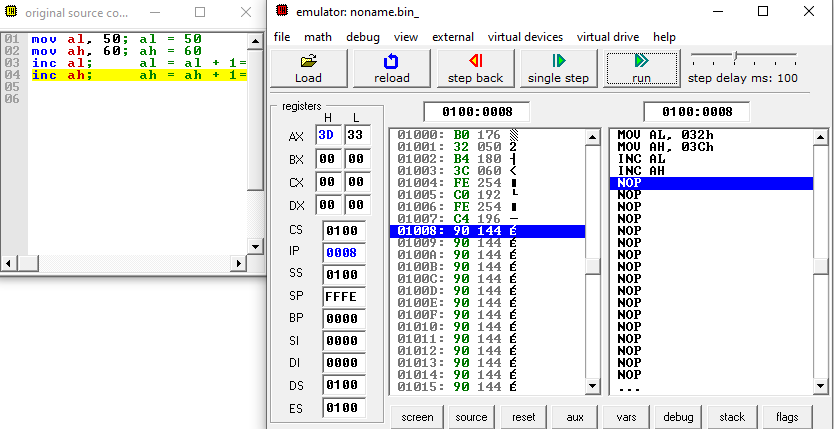


**mov al, 50; al = 50**

**mov ah, 60; ah = 60**

**inc al; al = al + 1= 50+1 = 51**

**inc ah; ah = ah + 1= 60 + 1 = 61**



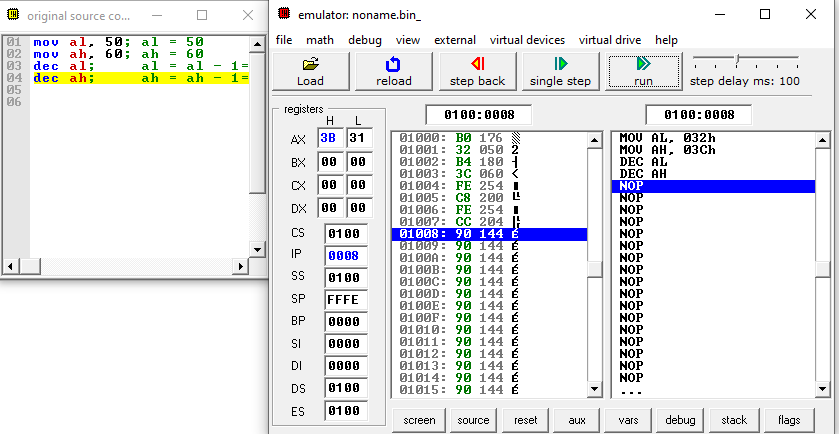
**DEC**: This instruction “DEC” is opposite of “INC”. It decreases the number by 1.

Example: **mov al, 50; al = 50**

**mov ah, 60; ah = 60**

**dec al; al = al - 1= 50 - 1 = 49**

**dec ah; ah = ah - 1= 60 - 1 = 59**

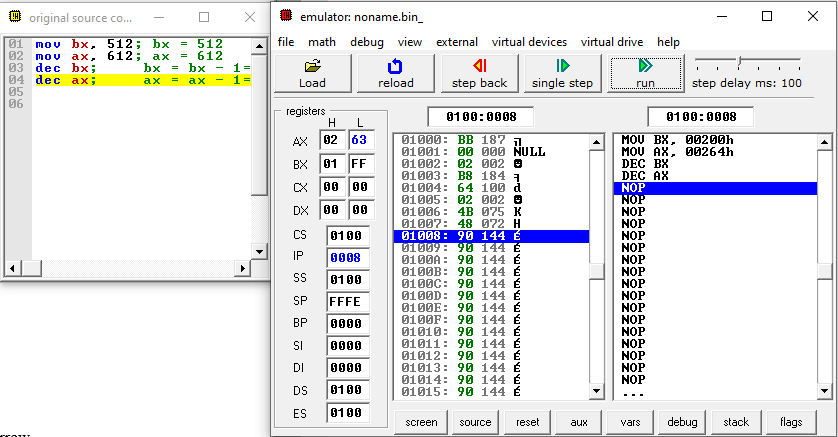


**mov bx, 512; bx = 512**

**mov ax, 612; ax = 612**

**dec bx; bx = bx - 1= 512 - 1 = 511**

**dec ax; ax = ax - 1= 612 - 1 = 611**



**SBB:** The instruction means Subtract with borrow. It subtracts the value with keeping the borrow into consiseration

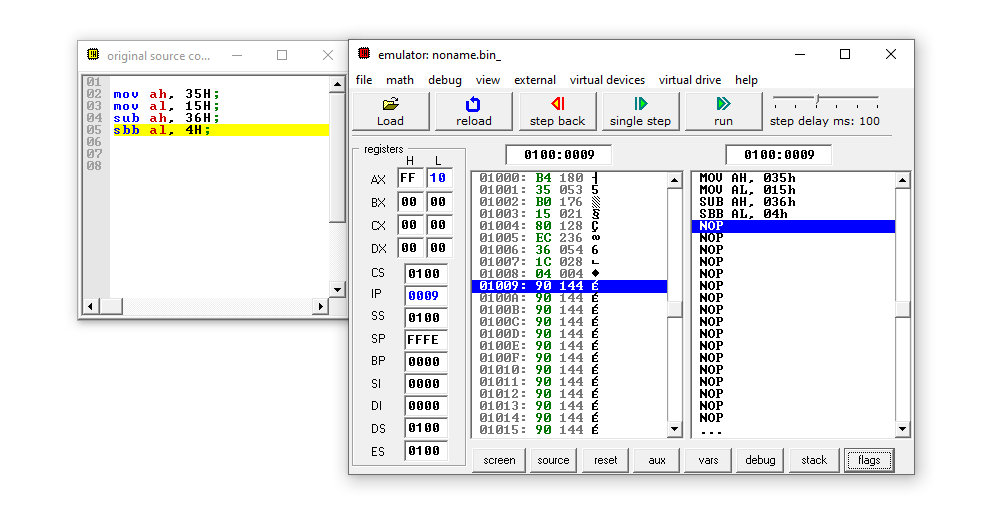
Example:

**mov ah, 35H;**

**mov al, 15H;**

**sub ah, 36H;**

**sbb al, 4H;**

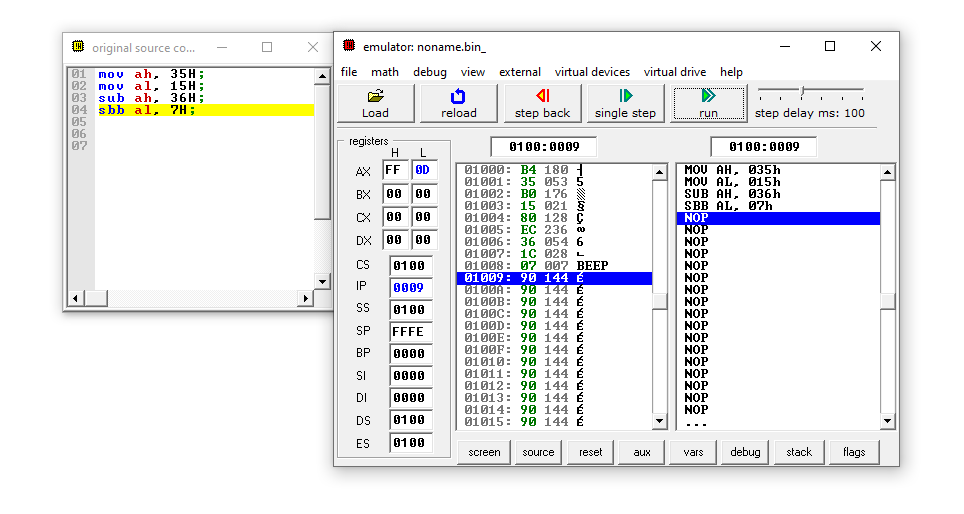


**mov ah, 35H;**

**mov al, 15H;**

**sub ah, 36H;**

**sbb al, 7H;**



Conclusion:

Each personal computer has a microprocessor that manages the computer's arithmetical, logical, and control activities. Each family of processors has its own set of instructions for handling various operations such as getting input from keyboard, displaying information on screen and performing various other jobs. These set of instructions are called 'machine language instructions'. A processor understands only machine language instructions, which are strings of 1's and 0's. However, machine language is too obscure and complex for using in software development. So, the low-level assembly language is designed for a specific family of processors that represents various instructions in symbolic code and a more understandable form.