Mustansiriyah University

College of Engineering

Electrical Engineering Department

(8086 Microprocessor Laboratory)

3rd Class

Contents:

No.	Subject	Page
1.	The 8086 Emulator	1-1
2A.	The Inside of 8086 Microprocessor & Move Instruction (Part 1)	2A -1
2B .	The Inside of 8086 Microprocessor & Move Instruction (Part 2)	2B-1
3.	Define Data	3-1
4.	Addressing Modes	4-1
5.	Arithmetic Instructions	5-1
6.	Logical Instructions	6-1
7.	Loop Instruction & Counter	7-1
8.	The Stack	8-1
9.	Applications 1	9-1
10.	Unsigned Jump Instructions	10-1
11.	Signed Conditional Jump Instructions	11-1
12	Applications 2	12-1
13.	Interrupt Service Routine	13-1

References

1) "THE INTEL MICROPROCESSORS 8086/8088, 80186/80188, 80286, 80386,80486, Pentium, Pentium Pro Processor, Pentium II,Pentium III, Pentium 4, and Core2with 64-Bit Extensions **Architecture,Programming, and Interfacing"-Eighth Edition BY: Barry B. Brey**

Experiment No: 1 - Part1 **Date:**

The 8086 Emulator

Object:

To learn how to use the 8086 emulator.

Theory:

The 8086 emulator is a SOFTWARE which by use it we can see how the program in 8086 microprocessor is run and what happened if this program is run.

After we initialize the SOFTWARE in the computer, double click on the SOFTWARE's icon we see the main window of the 8086 emulator (see Fig. (1-1)).

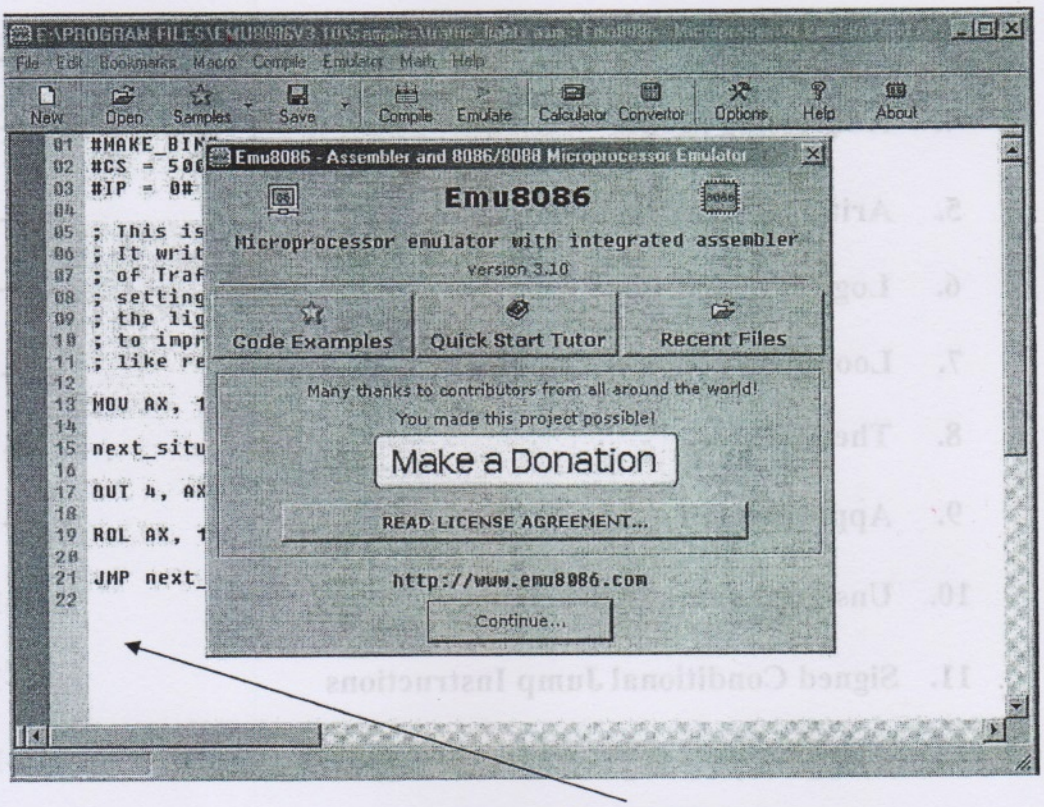

Fig. (1-1)
Main Window of 8086 Emulator

Click the (Continue...) icon and then click the (New) icon to open the new page (see Fig. (2-1))

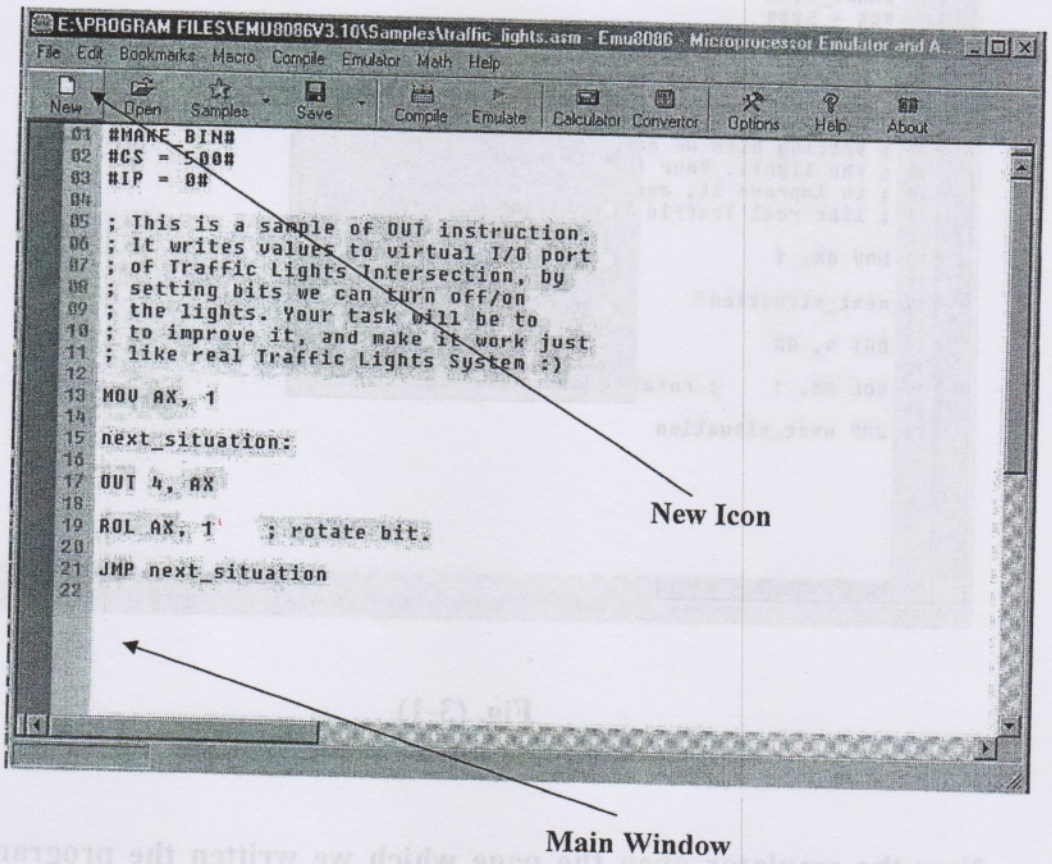

Fig. (2-1)
Main Window of 8086 Emulator

The (Choose Template for New File) window is open to select the file which you want to build it. There are four type of this files (Com, Exe, Bin, Boot template), now select the (Com Template) file and click OK (see Fig. (3-1)).

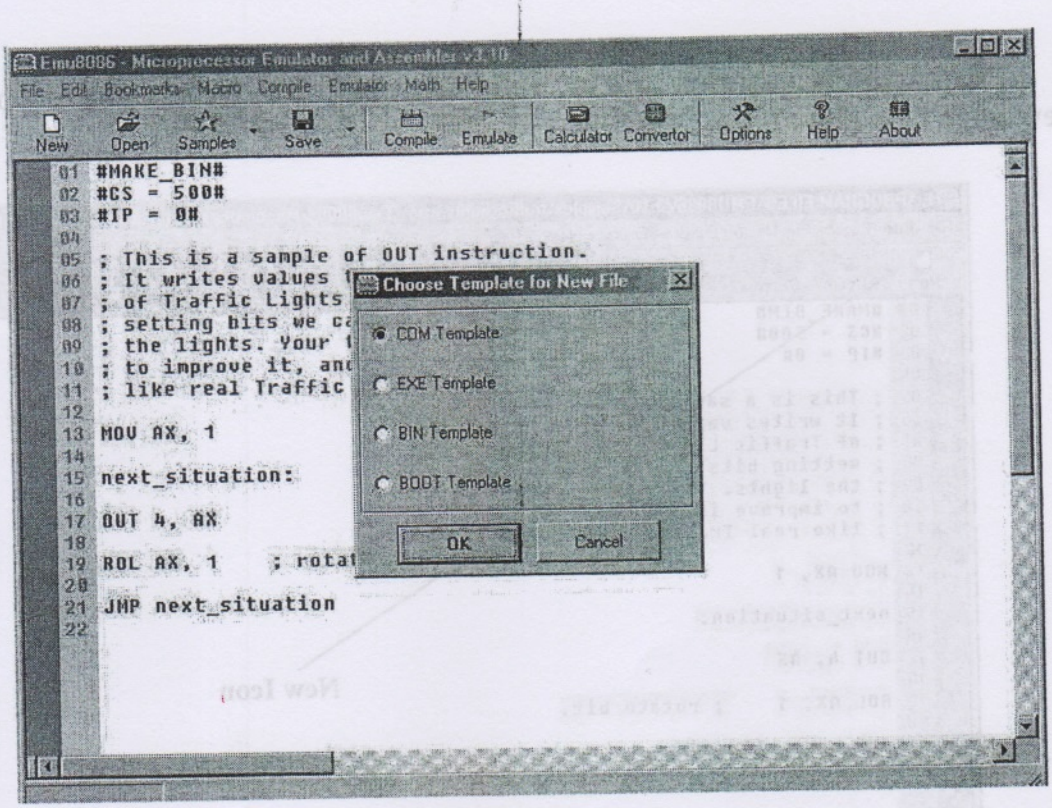

Fig. (3-1)
Choose Template for New File Window

Now the emulator open the page which we written the program on it this page is called the 8086 editor see Fig. (4-1).

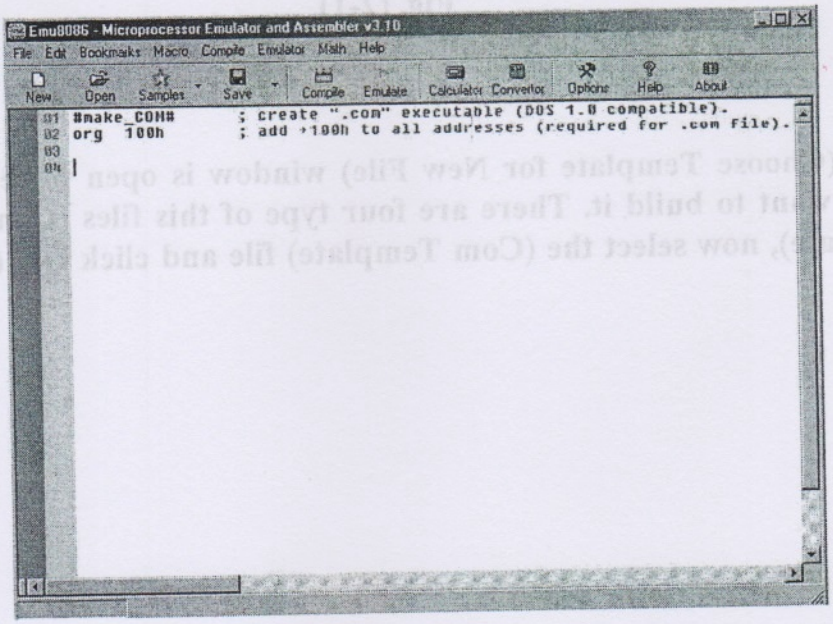

Fig. (4-1)
The 8086 Editor

Now after we written your program on the 8086 editor we need to run this program and see which component of the 8086 microprocessor is change. To run the program we click on the (Emulate) icon and the SOFTWARE is turn on the running window if the program is no syntax error.

Before us going on the running window we must know the important icons on the editor window and there are:

1. **New: To open a new editor widow.**
2. **Open: To open program which store on the computer.**
3. **Samples: To open the sample program.**
4. **Save: To store the written program on the computer.**
5. **Compile: To check the syntax error program.**
6. **Emulate: To turn on the running window**

Now we click on the Emulate icon and if the program which we written is no error the running page or running windows is appear see Fig. (5-1).

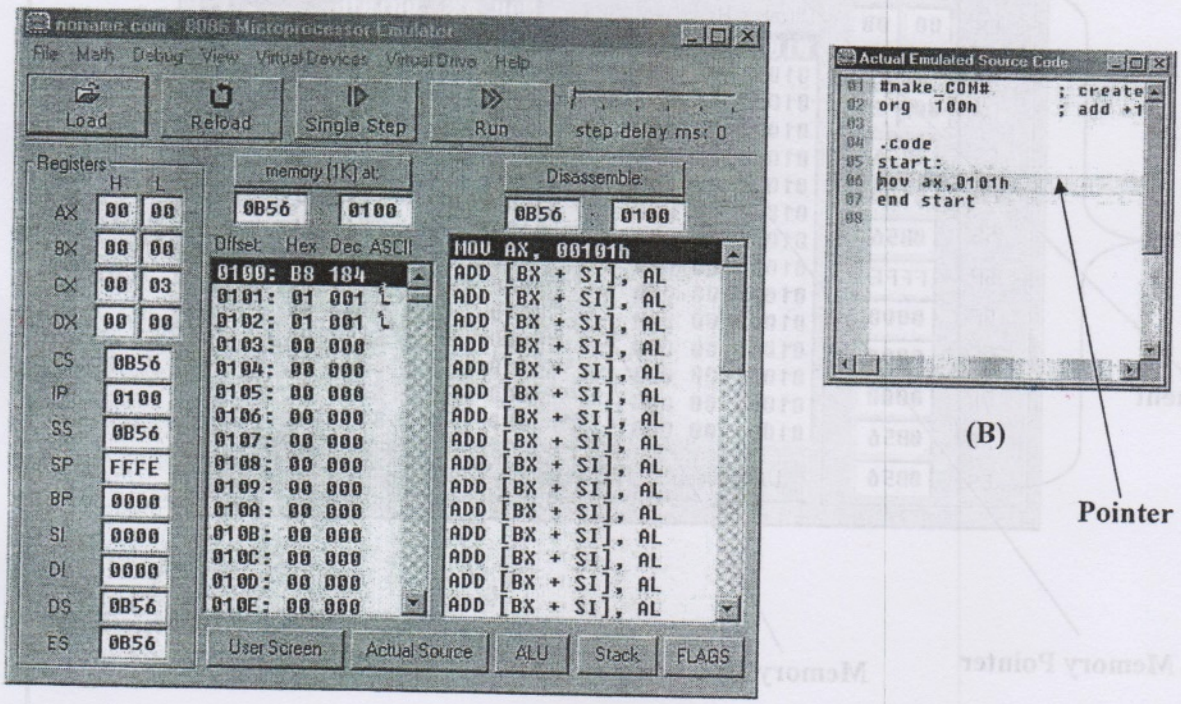

Fig. (5-1)
The 8086 Emulator-Running Window

Fig. (5-1-A) is called main running window and Fig. (5-1-B) is called the actual source code and this contain the program which is running in the CPU now and the pointer which is stopped on the instruction represent the instruction which is running now, above the pointer represents the instruction which is run before and down the pointer represent the instruction which is don't run yet.

Now we must define the component on the running window see Fig. (6-1) and this window contents:

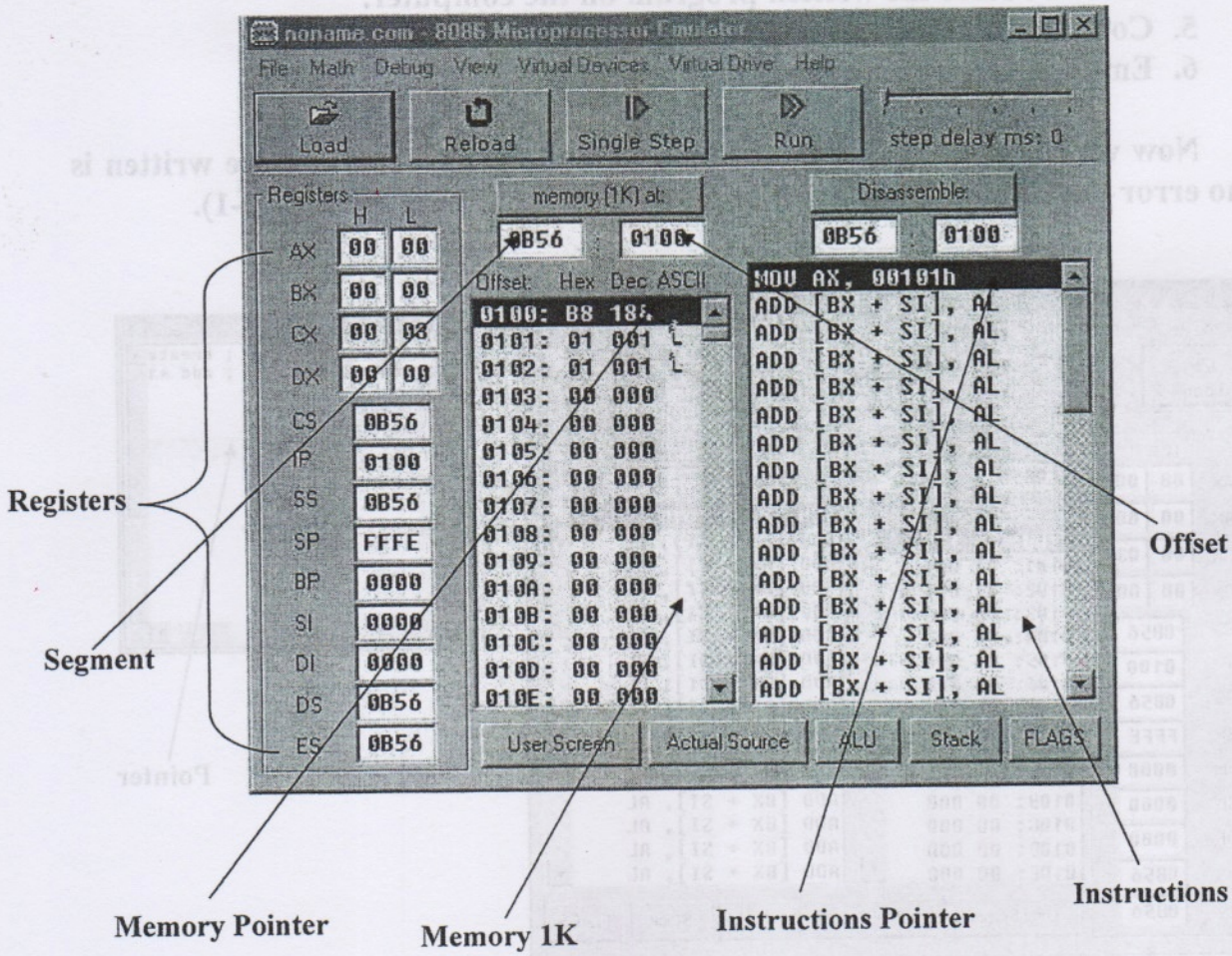

Fig. (6-1)
The 8086 Emulator Running Window

1. **Task Bar Icon:**
   A. **Load: To load the (com, exe, bin, boot) files to the CPU.**
   B. **Reload: To reload the same program to the CPU and reset the 8086 registers.**
   C. **Single Step: To run the program step by step.**
   D. **Run: To run all the instruction in the program.**

2. **Registers: Explain in the last experiment.**
3. **Memory: The 8086 microprocessor can access 1K bit of the memory.**
4. **Instruction: This is similar of the Fig. (5-1-B) but it represent the instructions in the memory.**
5. **Push Bottom Icon:**
   A. **User Screen: To see the screen if we have any printing instruction.**
   B. **Actual Source: To see the actual sources (Fig. (5-1-B).**
   C. **ALU: To see the Arithmetic Logic Unit of the 8986 microprocessor.**
   D. **Stack: To see the stack (Explain in the last experiment).**
   E. **Flag: To see the flag register (Explain in the last experiment).**

Experiment No: 1 part2 **Date:**

Memory Operations

Objective :

To write an assembly language program To Execute Data Transfer Instructions

Theory:

Data Read/Write process from /To Memory

Word Read

- Each of 1 MB memory address of 8086 represents a byte wide location
- 16bit words will be stored in two consecutive Memory location

- If first byte of the data is stored at an **even address** , 8086 can read the entire word in one operation.
  - For example if the 16 bit data is stored at even address 00520_H is 2607

    MOV BX, [00520]
    8086 reads the first byte and stores the data in BL and reads the 2nd byte and stores the data in BH

    BL← (00520)
    BH← (00521)

- If the first byte of the data is stored at an ODD address, 8086 needs two operation to read the 16 bit data
  - For example if the 16 bit data is stored at even address 00521_H is F520

    MOV BX, [00521]
    In first operation , 8086 reads the 16 bit data from the 00520 location and stores the data of 00521 location in register BL and discards the data of 00520 location

In 2nd operation, 8086 reads the 16 bit data from the 00522 location and stores the data of 00522 location in register BH and discards the data of 00523 location

BL← (00521)
BH← (00522)

Byte Read:

MOV BH, [Addr]

For Even Address:

Ex: MOV BH, [00520]

8086 reads the first byte from 00520 location and stores the data in BH and reads the 2nd byte from the 00521 location and ignores it

BH ← [00520]

For Odd Address

MOV BH, [Addr]

Ex: MOV BH, [00521]

8086 reads the first byte from 00520 location and ignores it and reads the 2nd byte from the 00521 location and stores the data in BH

BH ← [00521]

Data Transfer Between CPU and the Memory

Memory Write:

Byte Transfer: move BYTEPTR ds : [SI], 37H

Word Transfer: move WORDPTR ds : [SI], 1237H

Memory Read:

Byte Transfer: move al, BYTEPTR ds : [SI]

Transfers data from the physical memory address calculated using ds and [SI] to register AL (Lower byte of AX Register)

Word Transfer: move ax, WORDPTR ds : [SI]

Transfers data from the physical memory address calculated using ds and [SI] to register AL (Lower byte of AX Register) and the next byte from the next memory location calculated as ds:[SI +1] is transferred to AH (Higher byte of AX Register)

Memory operation through ax Register

Write:

MOV AX , 1234H
MOV WORDPTR ds: [SI], ax

Ds: 0000H
SI: 0500H
Physical Address: 00000+0500= 00500 H

The instruction transfers
34 → 00500H
12 → 00501H

Read:

MOV ax, WORDPTR ds: [SI]

Ds: 0000H
SI: 0500H
Physical Address: 00000+0500= 00500 H

The instruction transfers
AL ← (00500)
AH ← (00501)

Data Transfer Between CPU and the Port

Port addresses in 8086 are assigned either 8bit port address or 16 bit address

For a Port with 8bit port address:

Read Operation:

IN AL, Padr where Padr is the 8bit Port address

Ex: IN AL, 20 H

The instruction transfers data byte from the 8bit port address 20H to register AL

Write Operation:

OUT Padr, AL where Padr is the 8bit Port address

Ex: OUT 20 H, AL

The instruction transfers data byte from AL to the 8bit port address 20H .

For a Port with 16bit port address:

DX register is used to hold the Port address

Read Operation:

Example:

```
Mov DX, 4000H
IN al, DX
```

The instruction transfers data byte from 16bit port address 4000H contained in DX register to AL.

Write Operation:

Example:

```
MOV AL, 10H
MOV DX, 4000H
OUT DX, al
```

The instruction transfers data byte 10H from register AL to 16bit port address 4000H contained in DX

Procedures

1-write the following program

LABEL	ADDRESS	MNEMONICS	OPCODE	OPERAND	COMMENTS
START	1000H	MOV AX,FFFFH			Move the data FFFFH to AX reg
	1004H	MOV DX, 0012H			Move the data 0012H to DX reg
	1008H	MOV CX,01A1H			Move the Divisor 01A1 to CX reg
	100CH	DIV CX			Divide the content of AX&DX BY CX
	100EH	MOV[1200H],AX			Move the AX content to 1200H.
	1012H	MOV[1202H],DX			Move the DX content to 1202H.
	1016H	HLT			Stop the Process

Then give content of registers that will be changed for each instruction

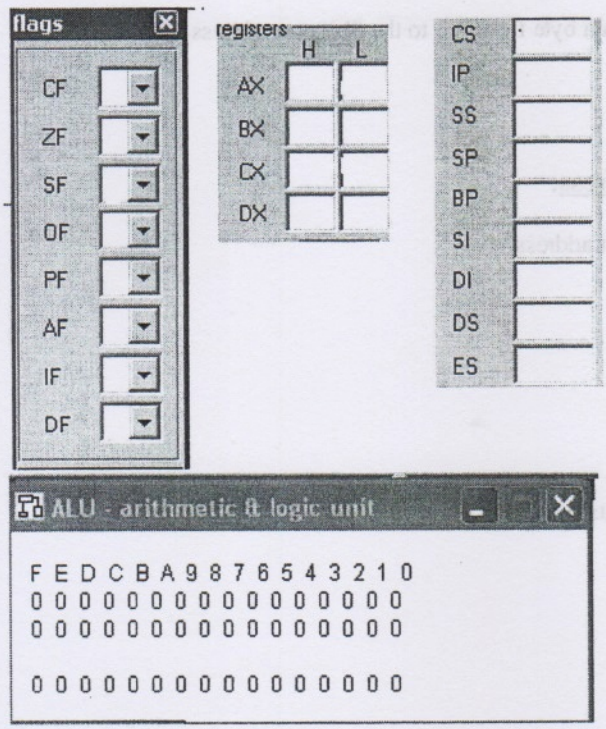

Fig (1-2) Registers and ALU -flags of 8086 microprocessor

2. Encode each of the instructions that follow into machine code.

a. MOV BX,OAAAA _____
b. MOV AX, BX _____
c. MOV AX, [BX] _____
d. MOV SI, 0300H _____
e. MOV AX, [0004] _____
f. MOV AX, [BX + SI] _____
g. MOV AX, [SI+4] _____
h. MOV AX, [BX + SI +4] _____

3-. How many bytes are required to store each of the machine code instructions in step 2:

a.
b.
c.
d.
e.
f.
g.
h.

3- Initialize the internal registers of the 80x86 as follows:

(AX) = 0000H
(BX) =0001H
(CX) = 0002H
(DX) = 0003H
(SI) = 0010H
(DI) = 0020H
(BP) = 0030H

Verify the initialization by displaying the new contents of the registers.

2. Fill all memory locations in the range DS: 00 through DS: 1F with 00H and then initialize the word storage locations that follow:

(DS: 0001H) = BBBBH
(DS: 0011H) = DDDDH
(DS: 0014H) = EEEEH
(DS: 0016H) = FFFFH

Home Work

1. **What are the results of the following program:**

```
MOV AL, 0AH
MOV BL, O6H
ADD AL, BL
MOV CL, AL
```

2. **What are the results of the following program:**

```
MOV AL, 44H
MOV AH, AL
MOV DX, 256
MOV BL, BH
MOV CH, 01000111B
MOV CL, 'A'
MOV DL, 174 O
MOV DH, DL
MOV DS, AX
MOV SS,BX
MOV ES, CX
```

3. **What are the data representation types used in the program above and what their letter indication.**
4. **Why is the better to use the registers to transform the data from and to the CPU over use the memory?**

5- **Encode each of the instructions that follow into machine code.**

   a. **MOV AX, BX; CS: 100;**
   b. **MOV AX, BX at location CS: 100**
   c. **MOVAX, OAAAA; CS: 110;**
   d. **MOV AX, [BX]; CS: 120;**
   e. **MOV AX, [4]; CS: 130;**
   f. **MOV AX, [BX + SI]; CS: 140;**
   g. **MOV AX, [SI +4]; CS: 150**
   h. **MOV AX, [BX+SI+4]; CS: 160;**