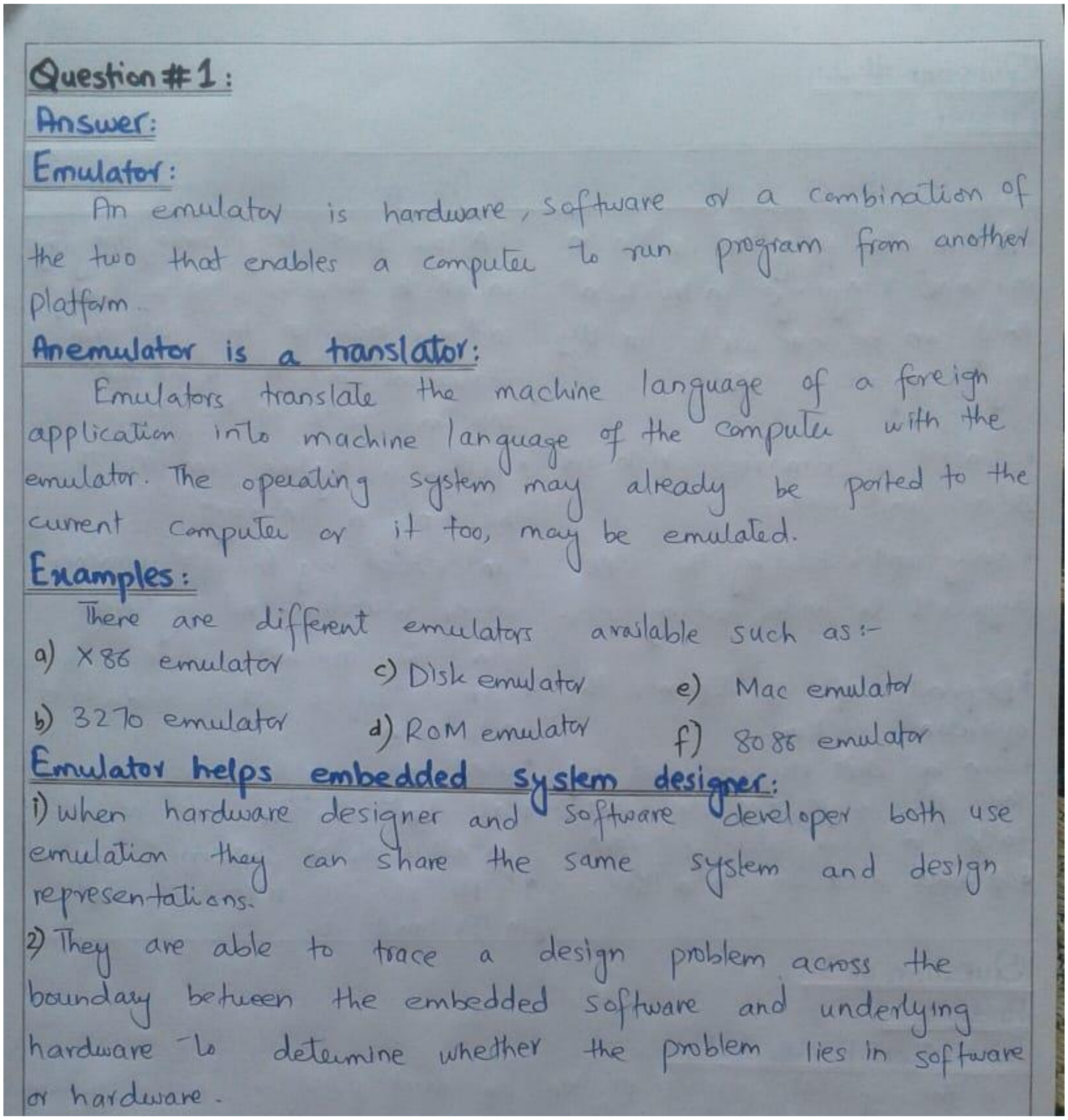
LAB MANUAL: 3

Describe briefly:

1. **What is Emulator and how it helps a hardware/embedded system designer?**



Question #1:

Answer:

Emulator:

An emulator is hardware, software or a combination of the two that enables a computer to run program from another platform.

Anemulator is a translator:

Emulators translate the machine language of a foreign application into machine language of the computer with the emulator. The operating system may already be ported to the current computer or it too, may be emulated.

Examples:

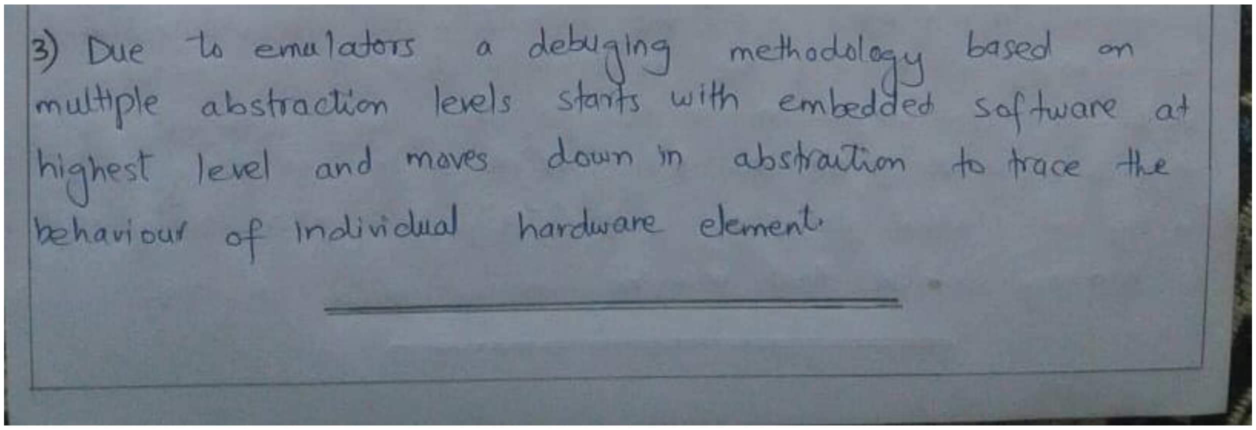
There are different emulators available such as:-

a) X86 emulator
b) 3270 emulator
c) Disk emulator
d) ROM emulator
e) Mac emulator
f) 8086 emulator

Emulator helps embedded system designer:

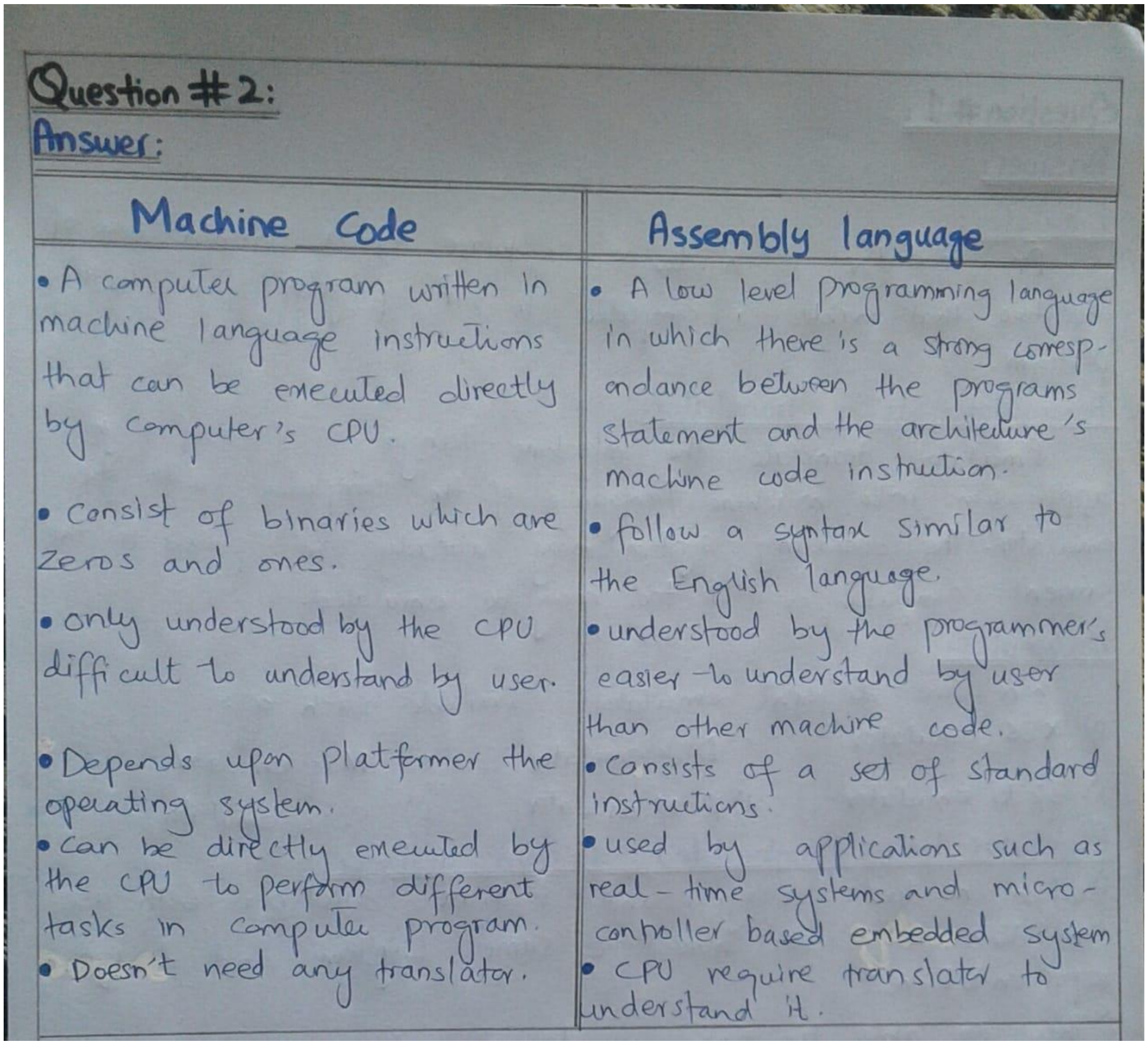
1) when hardware designer and software developer both use emulation they can share the same system and design representations.

2) They are able to trace a design problem across the boundary between the embedded software and underlying hardware to determine whether the problem lies in software or hardware.



3) Due to emulators a debuging methodology based on multiple abstraction levels starts with embedded software at highest level and moves down in abstraction to trace the behaviour of individual hardware element.

2. What is the difference between machine code and assembly language?

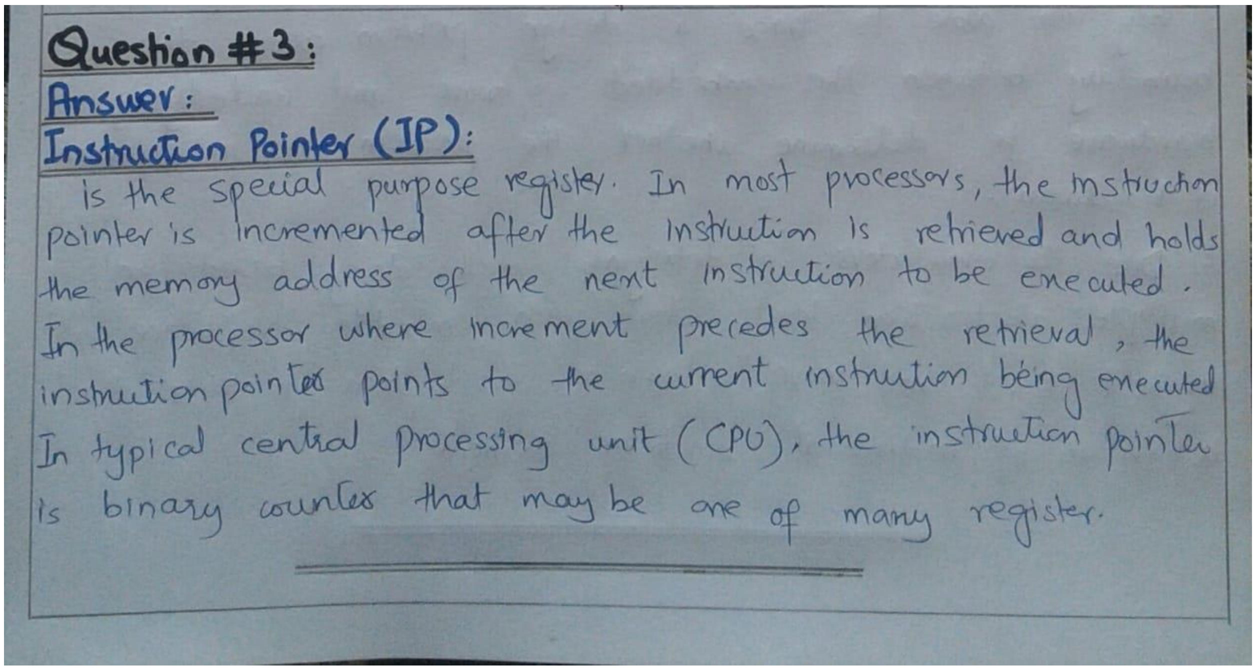


Question #2:

Answer:

Machine Code	Assembly language
• A computer program written in machine language instructions that can be executed directly by computer's CPU.	• A low level programming language in which there is a strong correspondance between the programs statement and the architecture's machine code instruction.
• Consist of binaries which are zeros and ones.	• follow a syntax similar to the English language.
• only understood by the CPU difficult to understand by user.	• understood by the programmer's easier to understand by user than other machine code.
• Depends upon platformer the operating system.	• Consists of a set of standard instructions.
• can be directly executed by the CPU to perform different tasks in computer program.	• used by applications such as real-time systems and micro-controller based embedded system
• Doesn't need any translator.	• CPU require translator to understand it.

3. Define Purpose of IP register?



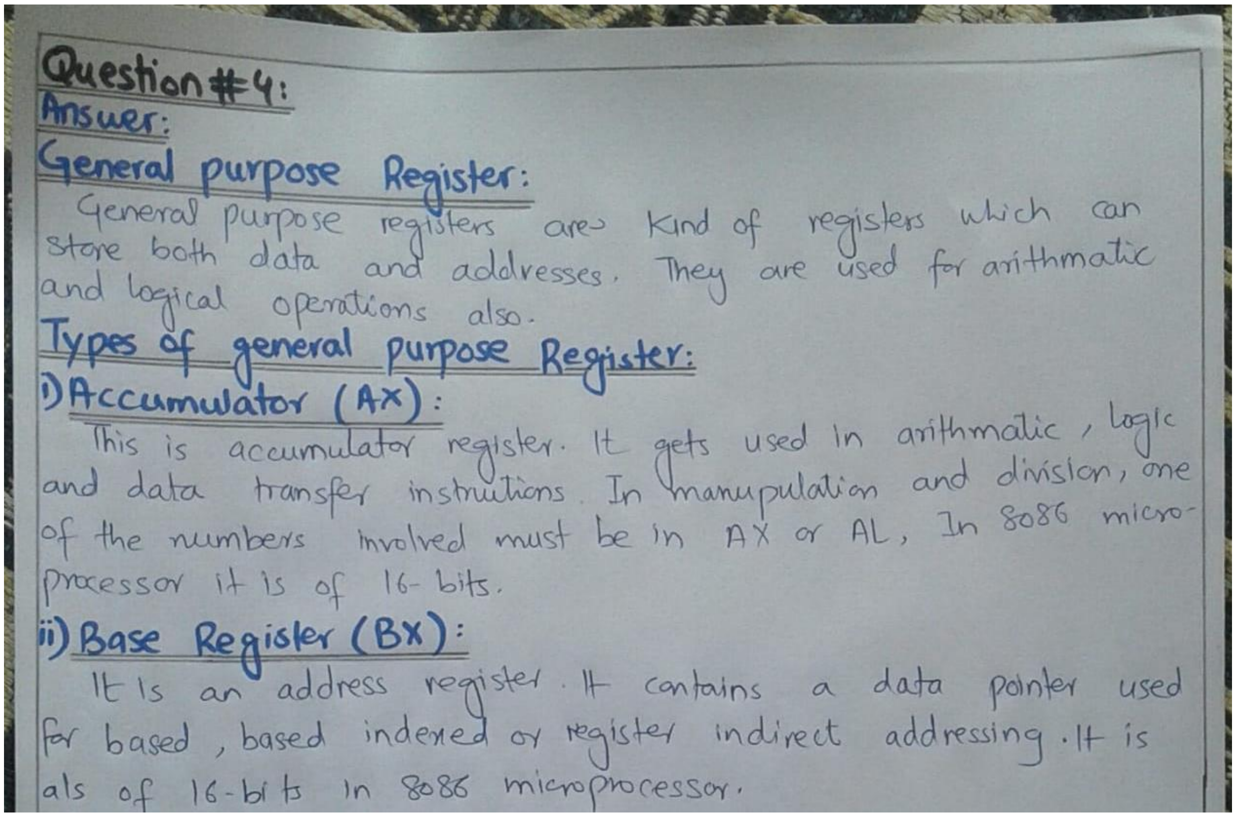
Question #3:

Answer:

Instruction Pointer (IP):

is the special purpose register. In most processors, the instruction pointer is incremented after the instruction is retrieved and holds the memory address of the next instruction to be executed. In the processor where increment precedes the retrieval, the instruction pointer points to the current instruction being executed. In typical central processing unit (CPU), the instruction pointer is binary counter that may be one of many register.

4. Write down the purpose of AX, BX, CX and DX Registers?



Question #4:

Answer:

General purpose Register:

General purpose registers are kind of registers which can store both data and addresses. They are used for arithmatic and logical operations also.

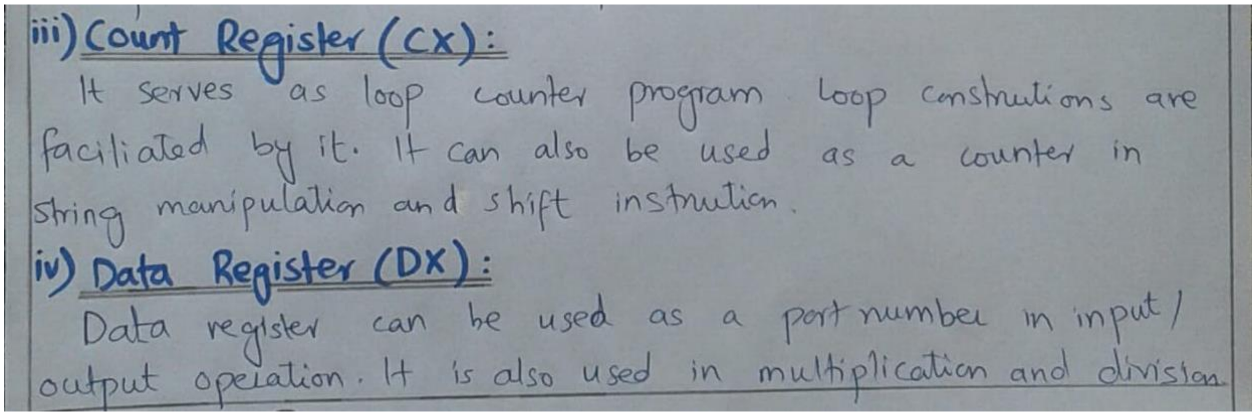
Types of general purpose Register:

i) Accumulator (AX):

This is accumulator register. It gets used in arithmatic, logic and data transfer instructions. In manupulation and division, one of the numbers involved must be in AX or AL, In 8086 micro-processor it is of 16-bits.

ii) Base Register (BX):

It is an address register. It contains a data pointer used for based, based indexed or register indirect addressing. It is als of 16-bits in 8086 microprocessor.



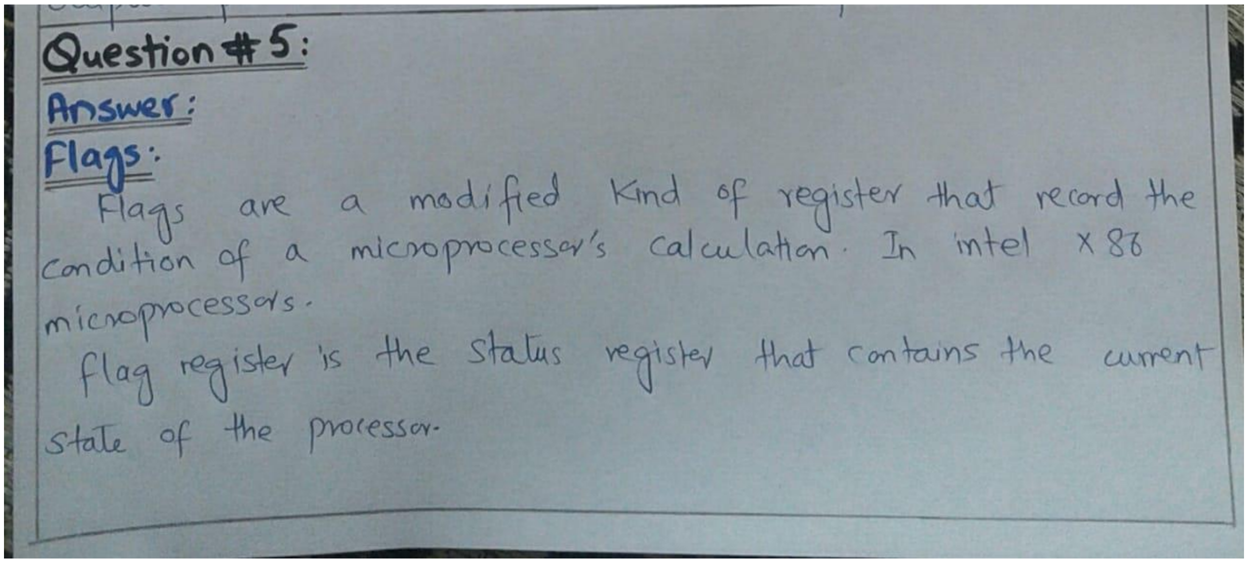
iii) Count Register (CX):

It serves as loop counter program loop constructions are faciliated by it. It can also be used as a counter in string manipulation and shift instruction.

iv) Data Register (DX):

Data register can be used as a port number in input/output operation. It is also used in multiplication and division.

5. How flags in a microprocessor help a hardware designer?



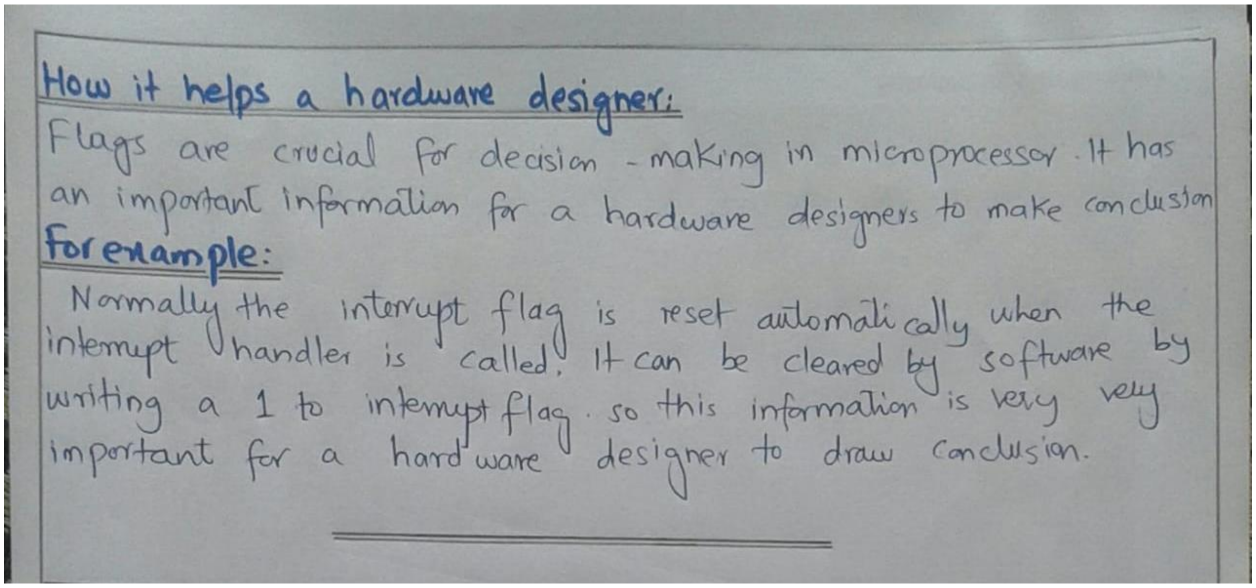
Question # 5:

Answer:

Flags:

Flags are a modified Kind of register that record the condition of a microprocessor's calculation. In intel x 86 microprocessors.

flag register is the Status register that contains the current state of the processor.



How it helps a hardware designer:

Flags are crucial for decision - making in microprocessor. It has an important information for a hardware designers to make conclusion

For example:

Normally the interrupt flag is reset automatically when the interrupt handler is called. It can be cleared by software by writing a 1 to interrupt flag. so this information is very very important for a hardware designer to draw conclusion.

LAB TASKS

TASK: 1

What effect on Conditional Flags will happens after the addition of 10110001 and 10101011? (Write state of each of the flag as observed, note values of flags after execution of every single instruction in the program).

SOURCE CODE:

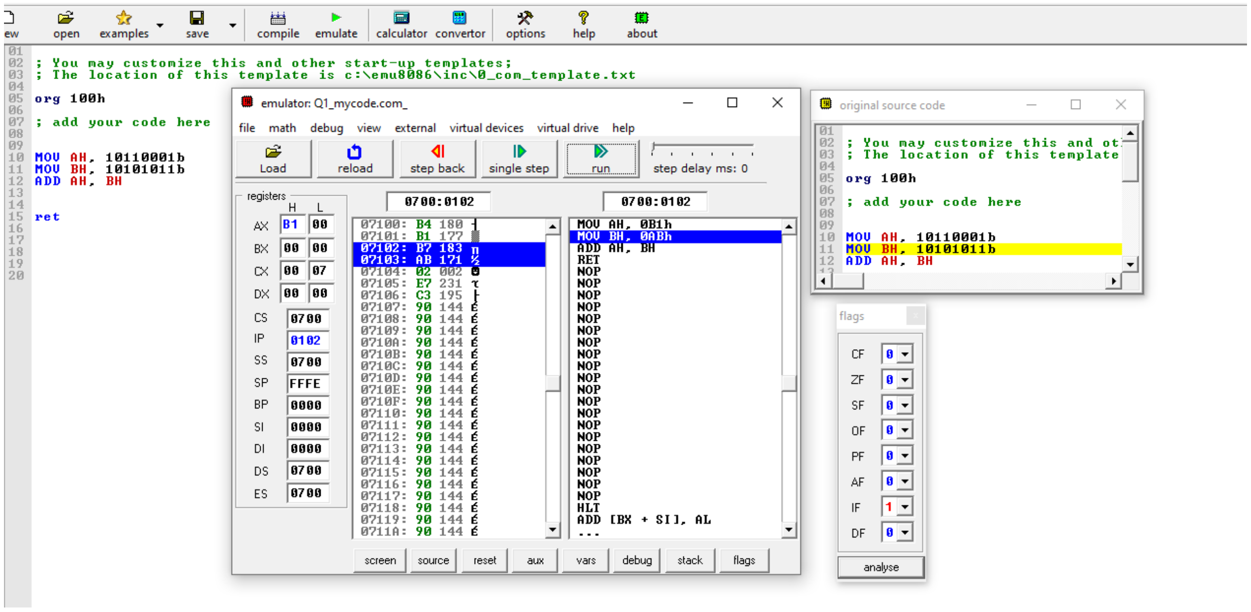
MOV AH, 10110001b

MOV BH, 10101011b

ADD AH, BH

Step#1:

This command is running 10110001b and is transferred to the AH registry. Carry Flag is zero after the first move



```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here


MOV AH, 10110001b
MOV BH, 10101011b
ADD AH, BH


ret
```

emulator: Q1_mycode.com_

registers

	H	L
AX	B1	00
BX	00	00
CX	00	07
DX	00	00

CS	0700
IP	0102
SS	0700
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

```
0700:0102
07100: B4 180 ┤
07101: B1 177 ▒
07102: B7 183 ╖
07103: AB 171 ½
07104: 02 002 ☻
07105: E7 231 τ
07106: C3 195 ├
07107: 90 144 É
07108: 90 144 É
07109: 90 144 É
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

```
0700:0102
MOV AH, 0B1h
MOV BH, 0ABh
ADD AH, BH
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
...
```

original source code

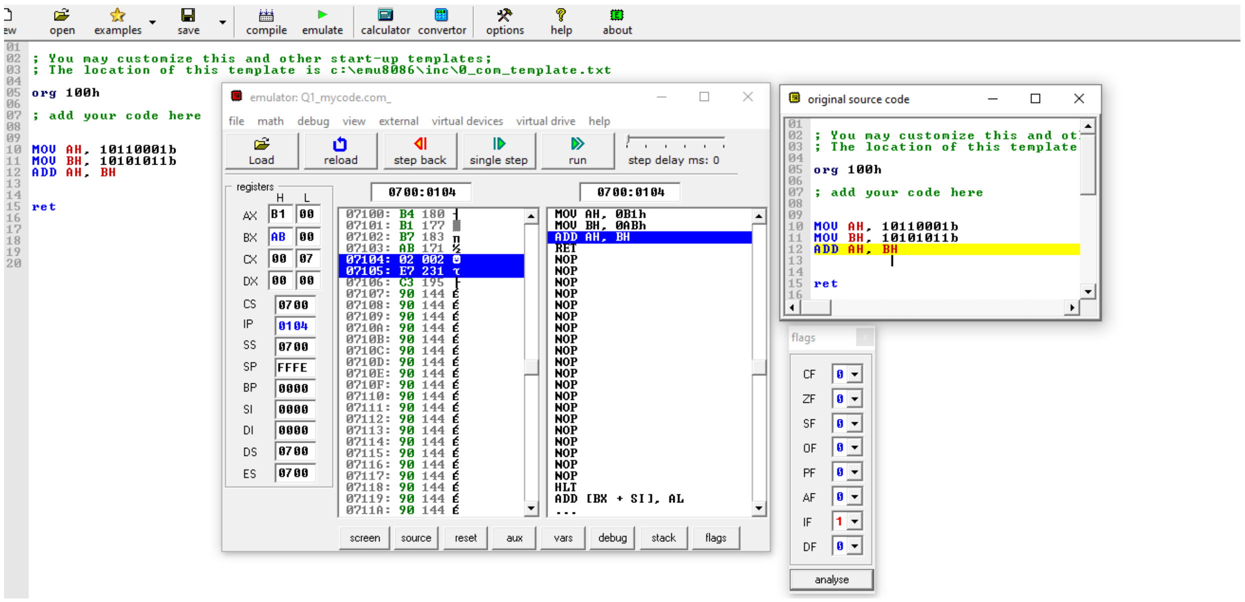
```
; You may customize this and ot
; The location of this template

org 100h

; add your code here


MOV AH, 10110001b
MOV BH, 10101011b
ADD AH, BH
```

flags

CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

Step#2:

This order is being executed 10101011b is transferred to the BH register. Carry Flag is zero after the second move.



```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here

MOV AH, 10110001b
MOV BH, 10101011b
ADD AH, BH

ret
```

emulator: Q1_mycode.com_

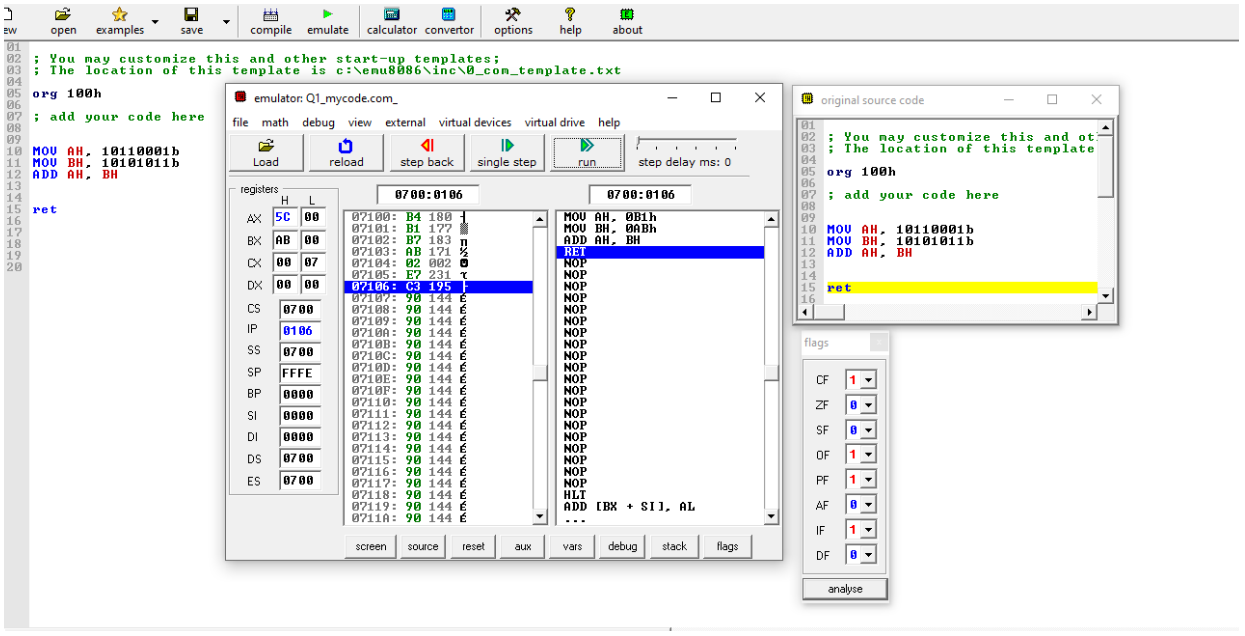
```
registers     0700:0104                  0700:0104
   H  L
AX B1 00      07100: B4 180 ┤            MOV AH, 0B1h
BX AB 00      07101: B1 177 ▒            MOV BH, 0ABh
CX 00 07      07102: B7 183 ╖            ADD AH, BH
DX 00 00      07103: AB 171 ½            RET
CS 0700       07104: 02 002 ☻            NOP
IP 0104       07105: E7 231 τ            NOP
SS 0700       07106: C3 195 ├            NOP
SP FFFE       07107: 90 144 É            NOP
BP 0000       07108: 90 144 É            NOP
SI 0000       07109: 90 144 É            NOP
DI 0000       0710A: 90 144 É            NOP
DS 0700       0710B: 90 144 É            NOP
ES 0700       0710C: 90 144 É            NOP
              0710D: 90 144 É            NOP
              0710E: 90 144 É            NOP
              0710F: 90 144 É            NOP
              07110: 90 144 É            NOP
              07111: 90 144 É            NOP
              07112: 90 144 É            NOP
              07113: 90 144 É            NOP
              07114: 90 144 É            NOP
              07115: 90 144 É            NOP
              07116: 90 144 É            NOP
              07117: 90 144 É            HLT
              07118: 90 144 É            ADD [BX + SI], AL
              07119: 90 144 É            ...
              0711A: 90 144 É
```

flags

```
CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0
```

Step#3:

The values in the AH are added to the values stored in the BH register. And save the result to the AX registry. After this phase, the Carry Flag is 1 because it keeps the carrying out after addition or borrows after subtraction. It also shows the state of mistake.



```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here

MOV AH, 10110001b
MOV BH, 10101011b
ADD AH, BH

ret
```

emulator: Q1_mycode.com_

```
registers     0700:0106                  0700:0106
   H  L
AX 5C 00      07100: B4 180 ┤            MOV AH, 0B1h
BX AB 00      07101: B1 177 ▒            MOV BH, 0ABh
CX 00 07      07102: B7 183 ╖            ADD AH, BH
DX 00 00      07103: AB 171 ½            RET
CS 0700       07104: 02 002 ☻            NOP
IP 0106       07105: E7 231 τ            NOP
SS 0700       07106: C3 195 ├            NOP
SP FFFE       07107: 90 144 É            NOP
BP 0000       07108: 90 144 É            NOP
SI 0000       07109: 90 144 É            NOP
DI 0000       0710A: 90 144 É            NOP
DS 0700       0710B: 90 144 É            NOP
ES 0700       0710C: 90 144 É            NOP
              0710D: 90 144 É            NOP
              0710E: 90 144 É            NOP
              0710F: 90 144 É            NOP
              07110: 90 144 É            NOP
              07111: 90 144 É            NOP
              07112: 90 144 É            NOP
              07113: 90 144 É            NOP
              07114: 90 144 É            NOP
              07115: 90 144 É            NOP
              07116: 90 144 É            NOP
              07117: 90 144 É            HLT
              07118: 90 144 É            ADD [BX + SI], AL
              07119: 90 144 É            ...
              0711A: 90 144 É
```

flags

```
CF 1
ZF 0
SF 0
OF 1
PF 1
AF 0
IF 1
DF 0
```

TASK: 2

Add two numbers in BIN, save the result in AX register and observe the value of flags. 1000 0000 0000 0000 1100 1000 0000 0000 (write state of each of the flag as observed, note values of flags after execution of every single instruction in the program.

SOURCE CODE:

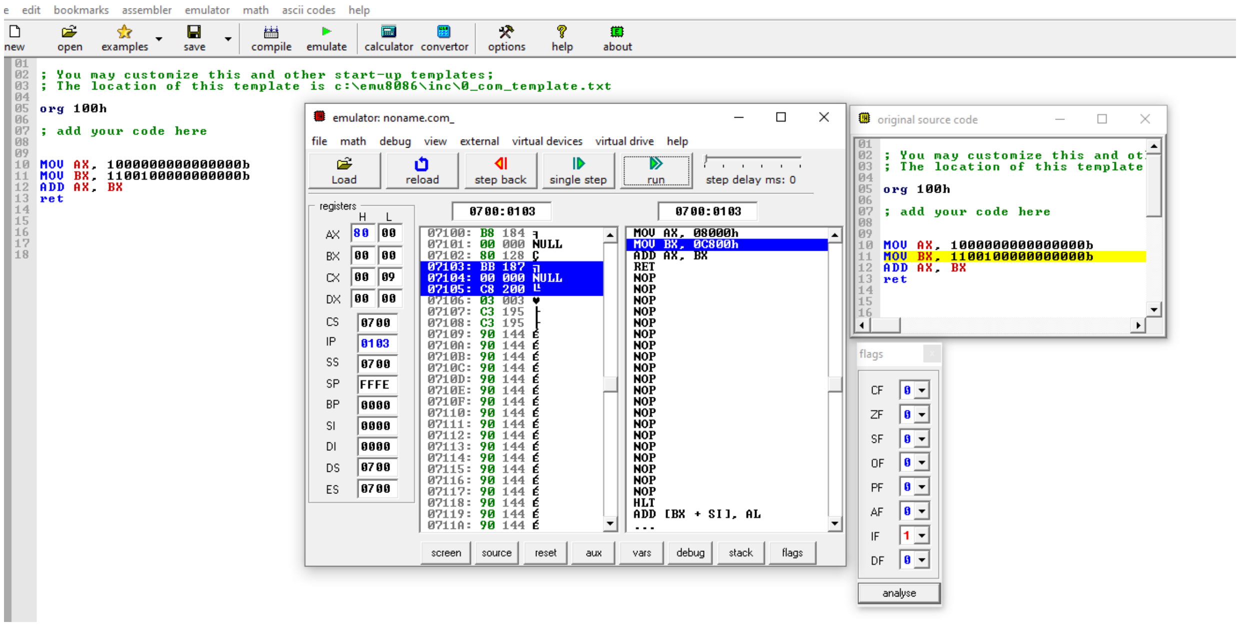
MOV AX, 1000 0000 0000 0000b

MOV BX, 1100 1000 0000 0000b

ADD AX, BX

Step#1:

This order is being executed 1000000000000B is shifted to the AX register. All the Flags are zero after the first move



```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here


MOV AX, 1000000000000000b
MOV BX, 1100100000000000b
ADD AX, BX
ret
```

emulator: noname.com_

registers	H	L
AX	80	00
BX	00	00
CX	00	09
DX	00	00

CS	0700
IP	0103
SS	0700
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

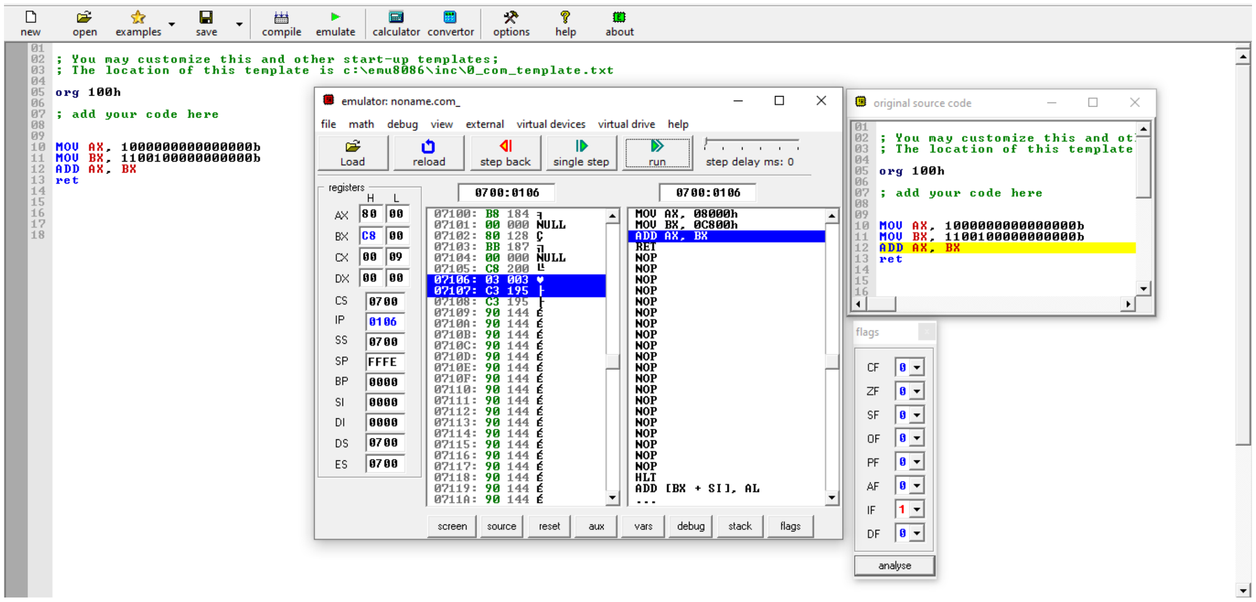
```
0700:0103
07100: B8 184 ╕
07101: 00 000 NULL
07102: 80 128 Ç
07103: BB 187 ╗
07104: 00 000 NULL
07105: C8 200 ╚
07106: 03 003 ♥
07107: C3 195 ├
07108: C3 195 ├
07109: 90 144 É
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

```
0700:0103
MOV AX, 08000h
MOV BX, 0C800h
ADD AX, BX
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
...
```

flags	
CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

Step#2:

This order is being executed 1100100000000000b is shifted to the BX register. All the Flags are zero after this move.



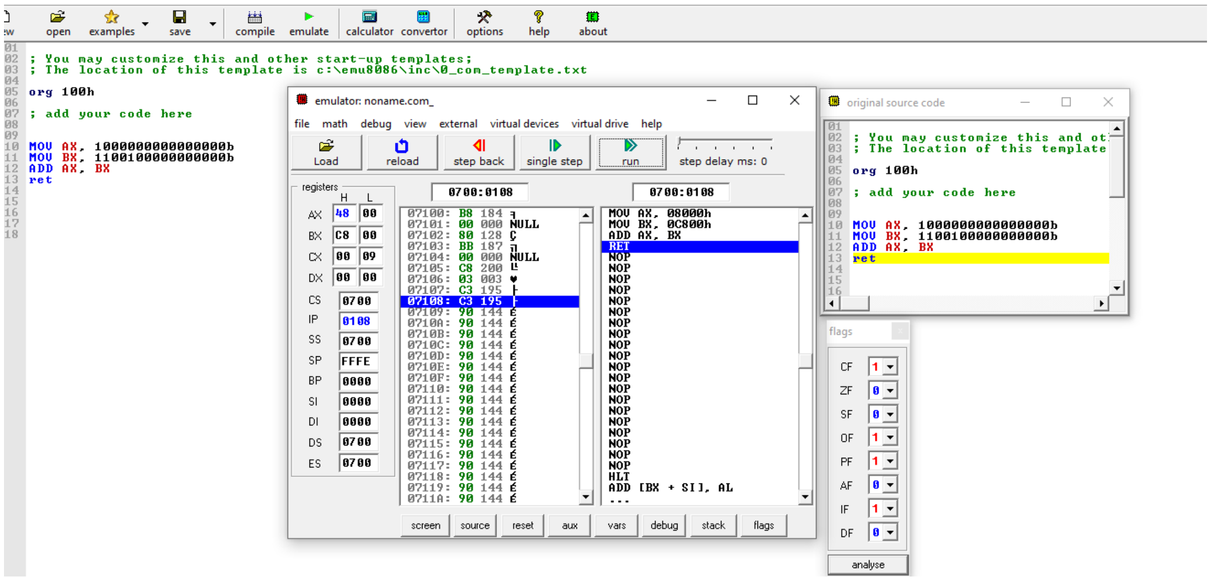
```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here


MOV AX, 1000000000000000b
MOV BX, 1100100000000000b
ADD AX, BX
ret
```

```
emulator: noname.com_

registers      0700:0106                 0700:0106
    H  L
AX  80 00      07100: B8 184 ╕           MOV AX, 08000h
BX  C8 00      07101: 00 000 NULL        MOV BX, 0C800h
CX  00 09      07102: 80 128 Ç           ADD AX, BX
DX  00 00      07103: BB 187 ╗           RET
CS  0700       07104: 00 000 NULL        NOP
IP  0106       07105: C8 200 ╚           NOP
SS  0700       07106: 03 003 ♥           NOP
SP  FFFE       07107: C3 195 ├           NOP
BP  0000       07108: C3 195 ├           NOP
SI  0000       07109: 90 144 É           NOP
DI  0000       0710A: 90 144 É           NOP
DS  0700       0710B: 90 144 É           NOP
ES  0700       0710C: 90 144 É           NOP
               0710D: 90 144 É           NOP
               0710E: 90 144 É           NOP
               0710F: 90 144 É           NOP
               07110: 90 144 É           NOP
               07111: 90 144 É           NOP
               07112: 90 144 É           NOP
               07113: 90 144 É           NOP
               07114: 90 144 É           NOP
               07115: 90 144 É           NOP
               07116: 90 144 É           NOP
               07117: 90 144 É           NOP
               07118: 90 144 É           HLT
               07119: 90 144 É           ADD [BX + SI], AL
               0711A: 90 144 É           ...
```

```
flags
CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0
```

Step#3:

Values in AX are added to the values stored in the BX register. And save the result in the AX registry. Carry Flag (CF) is one since it carries the carrying out after addition or borrows after subtraction. Often shows the state of mistake. Parity Flag (PF) is set to 1 because the system is even number one bit. After this third move, the Auxiliary Flag (AF) is set to 1 and keeps the carry (half)between bit positions 3 and 4 of the result after addition or the borrow after subtraction.



```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here


MOV AX, 1000000000000000b
MOV BX, 1100100000000000b
ADD AX, BX
ret
```

```
emulator: noname.com_

registers      0700:0108                 0700:0108
    H  L
AX  48 00      07100: B8 184 ╕           MOV AX, 08000h
BX  C8 00      07101: 00 000 NULL        MOV BX, 0C800h
CX  00 09      07102: 80 128 Ç           ADD AX, BX
DX  00 00      07103: BB 187 ╗           RET
CS  0700       07104: 00 000 NULL        NOP
IP  0108       07105: C8 200 ╚           NOP
SS  0700       07106: 03 003 ♥           NOP
SP  FFFE       07107: C3 195 ├           NOP
BP  0000       07108: C3 195 ├           NOP
SI  0000       07109: 90 144 É           NOP
DI  0000       0710A: 90 144 É           NOP
DS  0700       0710B: 90 144 É           NOP
ES  0700       0710C: 90 144 É           NOP
               0710D: 90 144 É           NOP
               0710E: 90 144 É           NOP
               0710F: 90 144 É           NOP
               07110: 90 144 É           NOP
               07111: 90 144 É           NOP
               07112: 90 144 É           NOP
               07113: 90 144 É           NOP
               07114: 90 144 É           NOP
               07115: 90 144 É           NOP
               07116: 90 144 É           NOP
               07117: 90 144 É           NOP
               07118: 90 144 É           HLT
               07119: 90 144 É           ADD [BX + SI], AL
               0711A: 90 144 É           ...
```

```
flags
CF 1
ZF 0
SF 0
OF 1
PF 1
AF 0
IF 1
DF 0
```

<u>TASK: 3</u>

Check out the status of flags for the following examples?

<u>Example: 1</u>

MOV DX, 126FH

ADD DX, 3465H

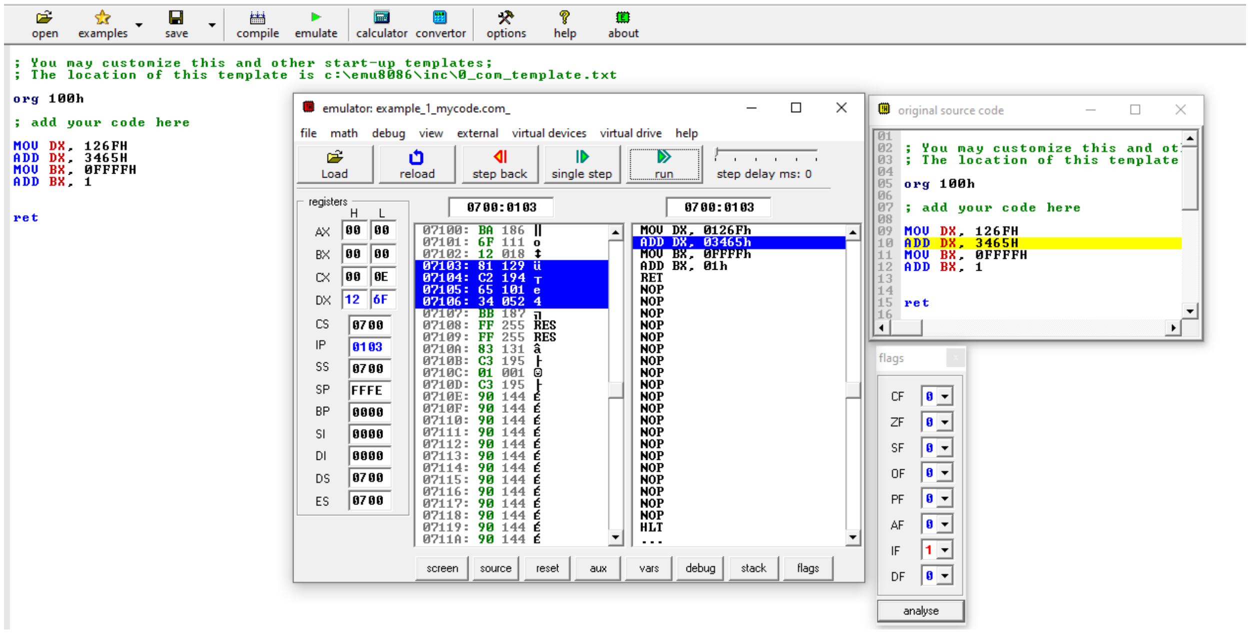
MOV BX, 0FFFFH

ADD BX, 1

Use Single Step and observe changes in flags after executing every single statement. Also, convert the numbers to binary and prove the results(manually) as observed using emulator.

<u>Step#1:</u>

This command is carried out. Hexadecimal 126FH is now in register DX. All of the Flags are zero after the first move.



```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here

MOV DX, 126FH
ADD DX, 3465H
MOV BX, 0FFFFH
ADD BX, 1


ret
```

emulator: example_1_mycode.com_

```
registers
     H   L
AX   00  00
BX   00  00
CX   00  0E
DX   12  6F
CS   0700
IP   0103
SS   0700
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
0700:0103

07100: BA 186 ║
07101: 6F 111 o
07102: 12 018 ↕
07103: 81 129 ü
07104: C2 194 ┬
07105: 65 101 e
07106: 34 052 4
07107: BB 187 ╗
07108: FF 255 RES
07109: FF 255 RES
0710A: 83 131 â
0710B: C3 195 ├
0710C: 01 001 ☺
0710D: C3 195 ├
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

```
0700:0103

MOV DX, 0126Fh
ADD DX, 03465h
MOV BX, 0FFFFh
ADD BX, 01h
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
...
```

original source code

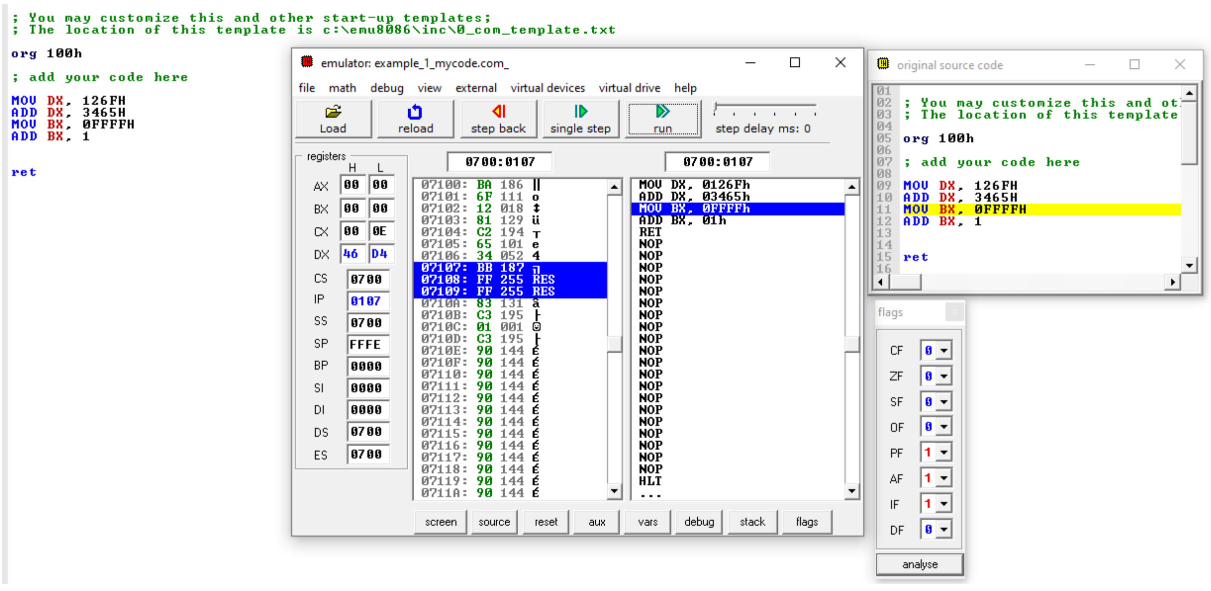
```
; You may customize this and ot
; The location of this template

org 100h

; add your code here

MOV DX, 126FH
ADD DX, 3465H
MOV BX, 0FFFFH
ADD BX, 1


ret
```

flags

```
CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0
```

Step#2:

In the values stored in register DX, the value 3465H is inserted. Since the device is an even number of one bit, the Parity Flag (PF) is set to 1 after this stage. Also set to 1, the Auxiliary Flag (AF) retains the carry (half carry) after addition or borrow after subtraction between bit positions 3 and 4 of the result.



```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here

MOV DX, 126FH
ADD DX, 3465H
MOV BX, 0FFFFH
ADD BX, 1


ret
```

```
    H  L
AX  00 00
BX  00 00
CX  00 0E
DX  46 D4
CS  0700
IP  0107
SS  0700
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0700
ES  0700
```

```
0700:0107

07100: BA 186 ╢
07101: 6F 111 o
07102: 12 018 ↕
07103: 81 129 ü
07104: C2 194 ┬
07105: 65 101 e
07106: 34 052 4
07107: BB 187 ╗
07108: FF 255 RES
07109: FF 255 RES
0710A: 83 131 â
0710B: C3 195 ├
0710C: 01 001 ☺
0710D: C3 195 ├
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

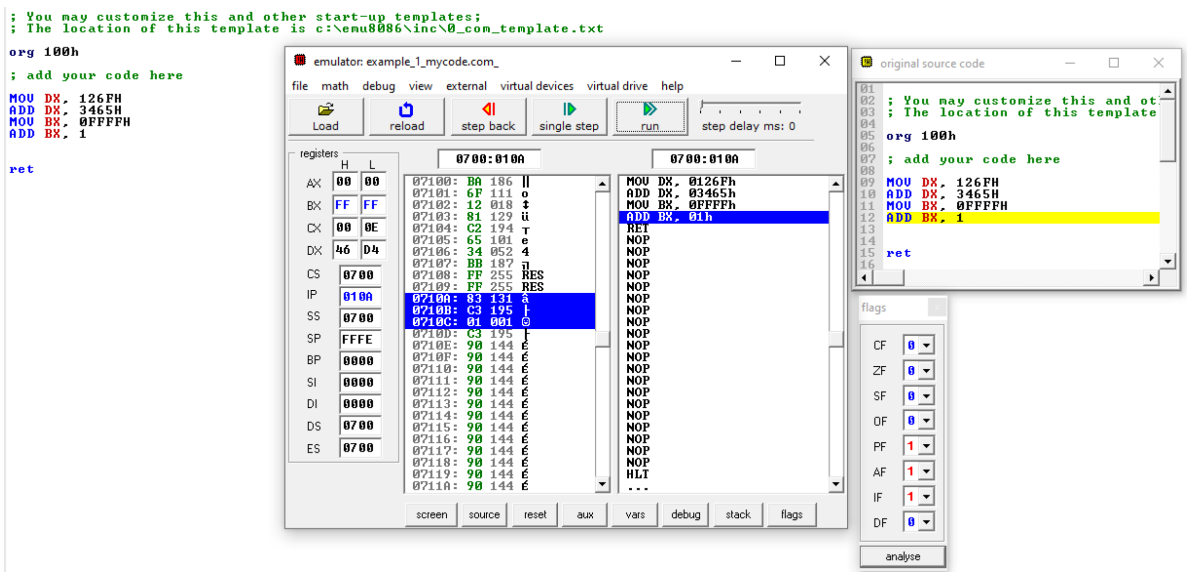
```
0700:0107

MOV DX, 0126Fh
ADD DX, 03465h
MOV BX, 0FFFFh
ADD BX, 01h
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
...
```

```
; You may customize this and ot
; The location of this template

org 100h

; add your code here

MOV DX, 126FH
ADD DX, 3465H
MOV BX, 0FFFFH
ADD BX, 1


ret
```

```
CF 0
ZF 0
SF 0
OF 0
PF 1
AF 1
IF 1
DF 0
```

Step#3:

This order will be executed 0FFFFH Hexadecimal is shifted to the BX register. After the first stage, the Parity Flag (PF) is set to 1 because the device is even number one bit. And the Auxiliary Flag (AF) is also set to 1 where the carrying retains the carrying (half-carry) after addition or the borrowing after subtraction between bit positions 3 and 4 of the result.



```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here

MOV DX, 126FH
ADD DX, 3465H
MOV BX, 0FFFFH
ADD BX, 1

ret
```

emulator: example_1_mycode.com_

```
registers
     H   L
AX   00  00
BX   FF  FF
CX   00  0E
DX   46  D4
CS   0700
IP   010A
SS   0700
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
0700:010A                0700:010A
07100: BA 186 ╗          MOV DX, 0126Fh
07101: 6F 111 o          ADD DX, 03465h
07102: 12 018 ↕          MOV BX, 0FFFFh
07103: 81 129 ü          ADD BX, 01h
07104: C2 194 ┬          RET
07105: 65 101 e          NOP
07106: 34 052 4          NOP
07107: BB 187 ╗          NOP
07108: FF 255 RES        NOP
07109: FF 255 RES        NOP
0710A: 83 131 â          NOP
0710B: C3 195 ├          NOP
0710C: 01 001 ☺          NOP
0710D: C3 195 ├          NOP
0710E: 90 144 É          NOP
0710F: 90 144 É          NOP
07110: 90 144 É          NOP
07111: 90 144 É          NOP
07112: 90 144 É          NOP
07113: 90 144 É          NOP
07114: 90 144 É          NOP
07115: 90 144 É          NOP
07116: 90 144 É          NOP
07117: 90 144 É          NOP
07118: 90 144 É          NOP
07119: 90 144 É          HLT
0711A: 90 144 É          ...
```

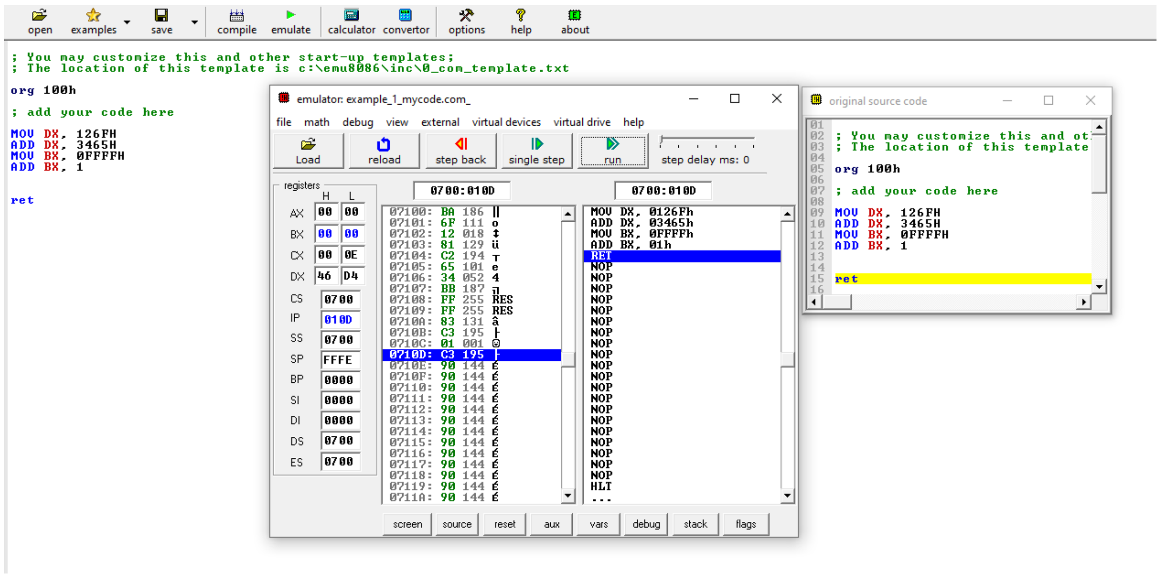
original source code

```
; You may customize this and ot
; The location of this template

org 100h

; add your code here

MOV DX, 126FH
ADD DX, 3465H
MOV BX, 0FFFFH
ADD BX, 1


ret
```

```
flags
CF 0
ZF 0
SF 0
OF 0
PF 1
AF 1
IF 1
DF 0
```

Step#4:

The value 1H is applied to the values stored in the BX register. After this phase, the Carry Flag (CF) is one because it keeps the carry out after addition or borrows after subtraction. Often shows the state of mistake. Parity Flag (PF) is set to 1 because the system is even number one bit. And the Auxiliary Flag (AF) is also set to 1 holding the carry holding the carry (half-carry) after addition or the borrow after subtraction between bit positions 3 and 4 of the result.



```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here

MOV DX, 126FH
ADD DX, 3465H
MOV BX, 0FFFFH
ADD BX, 1

ret
```

emulator: example_1_mycode.com_

```
registers
     H   L
AX   00  00
BX   00  00
CX   00  0E
DX   46  D4
CS   0700
IP   010D
SS   0700
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
0700:010D                0700:010D
07100: BA 186 ╗          MOV DX, 0126Fh
07101: 6F 111 o          ADD DX, 03465h
07102: 12 018 ↕          MOV BX, 0FFFFh
07103: 81 129 ü          ADD BX, 01h
07104: C2 194 ┬          RET
07105: 65 101 e          NOP
07106: 34 052 4          NOP
07107: BB 187 ╗          NOP
07108: FF 255 RES        NOP
07109: FF 255 RES        NOP
0710A: 83 131 â          NOP
0710B: C3 195 ├          NOP
0710C: 01 001 ☺          NOP
0710D: C3 195 ├          NOP
0710E: 90 144 É          NOP
0710F: 90 144 É          NOP
07110: 90 144 É          NOP
07111: 90 144 É          NOP
07112: 90 144 É          NOP
07113: 90 144 É          NOP
07114: 90 144 É          NOP
07115: 90 144 É          NOP
07116: 90 144 É          NOP
07117: 90 144 É          NOP
07118: 90 144 É          NOP
07119: 90 144 É          HLT
0711A: 90 144 É          ...
```

original source code

```
; You may customize this and ot
; The location of this template

org 100h

; add your code here

MOV DX, 126FH
ADD DX, 3465H
MOV BX, 0FFFFH
ADD BX, 1


ret
```

Example: 2

MOV BL, +8

MOV DH, +4

ADD BL, DH

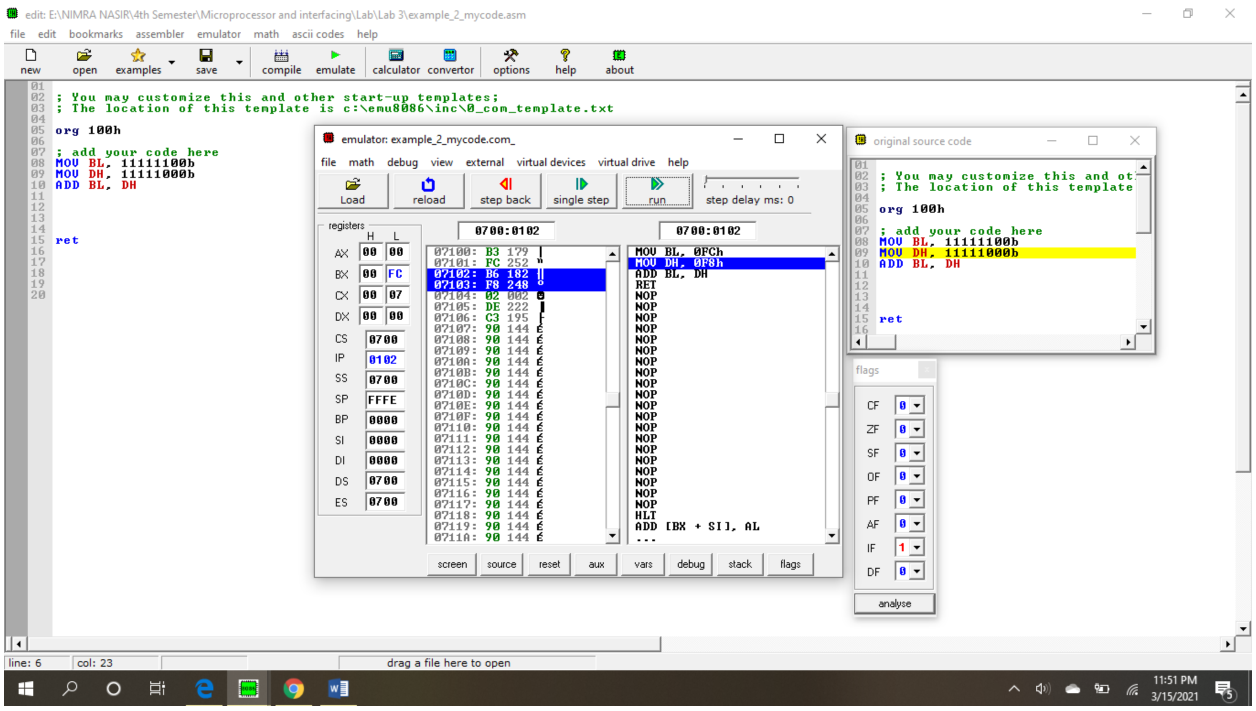
SOURCE CODE:

Step#1:

In the first step I have converted the + 8 and +4 in binary according to formula As we know that in case of (+)(Decimal value) we just convert simply the decimal value first into the binary digits then we place 0 for + sign to the left most position of binary no: So, +8=01000 b For 8 bits binary no the +8= 00001000b Same for the +4: +4= 00000100b Now The above code becomes: MOV BL, 00001000b MOV DH, 00000100b ADD BL,DH

Step#2:

This command is executed 00001000b is moved to register BL. After first step all the Flags are zero.


edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_2_mycode.asm

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV BL, 11111100b
MOV DH, 11111000b
ADD BL, DH

ret
```

emulator: example_2_mycode.com_

```
registers   H   L
AX          00  00
BX          00  FC
CX          00  07
DX          00  00
CS  0700
IP  0102
SS  0700
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0700
ES  0700
```

```
0700:0102                 0700:0102
07100: B3 179             MOV BL, 0FCh
07101: FC 252             MOV DH, 0F8h
07102: B6 182             ADD BL, DH
07103: F8 248             RET
07104: 02 002             NOP
07105: DE 222             NOP
07106: C3 195             NOP
07107: 90 144 É           NOP
07108: 90 144 É           NOP
07109: 90 144 É           NOP
0710A: 90 144 É           NOP
0710B: 90 144 É           NOP
0710C: 90 144 É           NOP
0710D: 90 144 É           NOP
0710E: 90 144 É           NOP
0710F: 90 144 É           NOP
07110: 90 144 É           NOP
07111: 90 144 É           NOP
07112: 90 144 É           NOP
07113: 90 144 É           NOP
07114: 90 144 É           NOP
07115: 90 144 É           NOP
07116: 90 144 É           NOP
07117: 90 144 É           NOP
07118: 90 144 É           HLT
07119: 90 144 É           ADD [BX + SI], AL
0711A: 90 144 É           ...
```

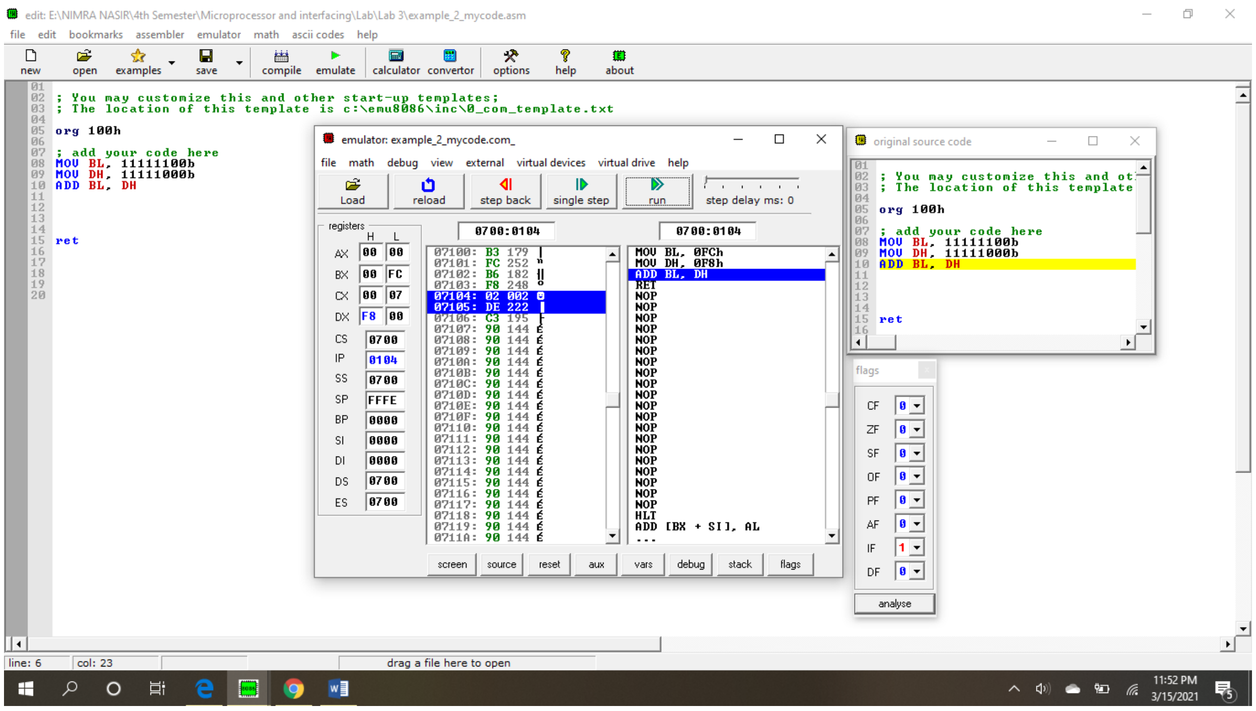
flags: CF 0, ZF 0, SF 0, OF 0, PF 0, AF 0, IF 1, DF 0

line: 6 col: 23 drag a file here to open

11:51 PM 3/15/2021

Step#3:

This command is executed 00000100b is moved to register DH. After second step all the Flags are zero.


edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_2_mycode.asm

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV BL, 11111100b
MOV DH, 11111000b
ADD BL, DH

ret
```

emulator: example_2_mycode.com_

```
registers   H   L
AX          00  00
BX          00  FC
CX          00  07
DX          F8  00
CS  0700
IP  0104
SS  0700
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0700
ES  0700
```

```
0700:0104                 0700:0104
07100: B3 179             MOV BL, 0FCh
07101: FC 252             MOV DH, 0F8h
07102: B6 182             ADD BL, DH
07103: F8 248             RET
07104: 02 002             NOP
07105: DE 222             NOP
07106: C3 195             NOP
07107: 90 144 É           NOP
07108: 90 144 É           NOP
07109: 90 144 É           NOP
0710A: 90 144 É           NOP
0710B: 90 144 É           NOP
0710C: 90 144 É           NOP
0710D: 90 144 É           NOP
0710E: 90 144 É           NOP
0710F: 90 144 É           NOP
07110: 90 144 É           NOP
07111: 90 144 É           NOP
07112: 90 144 É           NOP
07113: 90 144 É           NOP
07114: 90 144 É           NOP
07115: 90 144 É           NOP
07116: 90 144 É           NOP
07117: 90 144 É           NOP
07118: 90 144 É           HLT
07119: 90 144 É           ADD [BX + SI], AL
0711A: 90 144 É           ...
```

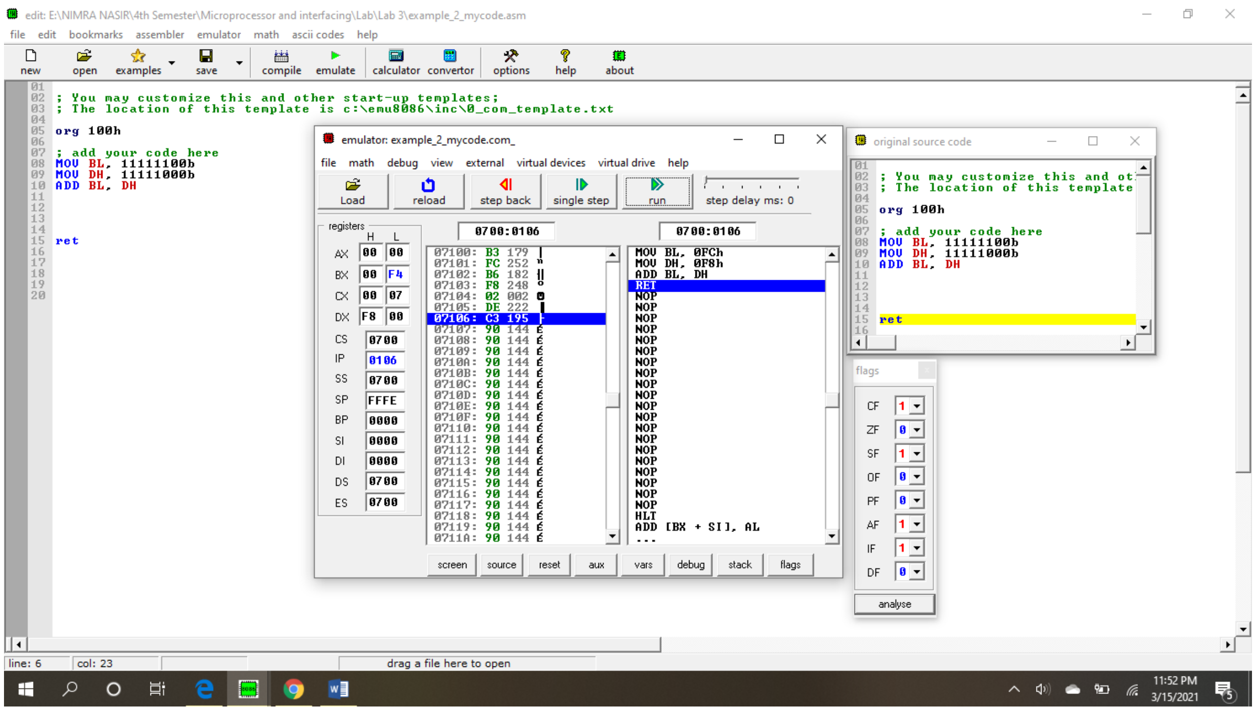
flags: CF 0, ZF 0, SF 0, OF 0, PF 0, AF 0, IF 1, DF 0

line: 6 col: 23 drag a file here to open

11:52 PM 3/15/2021

Step#4:

After this step DH Value will be added to the BL Register value. And As shown in the following figure the parity flag is set to 1 because system is even number one bit


edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_2_mycode.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
emulator: example_2_mycode.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags
original source code
analyse
line: 6 col: 23
drag a file here to open
11:52 PM
3/15/2021

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV BL, 11111100b
MOV DH, 11111000b
ADD BL, DH




ret
```

```
registers
      H    L
AX   00   00
BX   00   F4
CX   00   07
DX   F8   00
CS   0700
IP   0106
SS   0700
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
0700:0106                      0700:0106
07100: B3 179 ┃                MOV BL, 0FCh
07101: FC 252 ⁿ                MOV DH, 0F8h
07102: B6 182 ╢                ADD BL, DH
07103: F8 248 °                RET
07104: 02 002 ☻                NOP
07105: DE 222 ▐                NOP
07106: C3 195 ├                NOP
07107: 90 144 É                NOP
07108: 90 144 É                NOP
07109: 90 144 É                NOP
0710A: 90 144 É                NOP
0710B: 90 144 É                NOP
0710C: 90 144 É                NOP
0710D: 90 144 É                NOP
0710E: 90 144 É                NOP
0710F: 90 144 É                NOP
07110: 90 144 É                NOP
07111: 90 144 É                NOP
07112: 90 144 É                NOP
07113: 90 144 É                NOP
07114: 90 144 É                NOP
07115: 90 144 É                NOP
07116: 90 144 É                NOP
07117: 90 144 É                NOP
07118: 90 144 É                HLT
07119: 90 144 É                ADD [BX + SI], AL
0711A: 90 144 É                ...
```

```
flags
CF 1
ZF 0
SF 1
OF 0
PF 0
AF 1
IF 1
DF 0
```

Example: 3

MOV AL, +66d

MOV CL, +69d

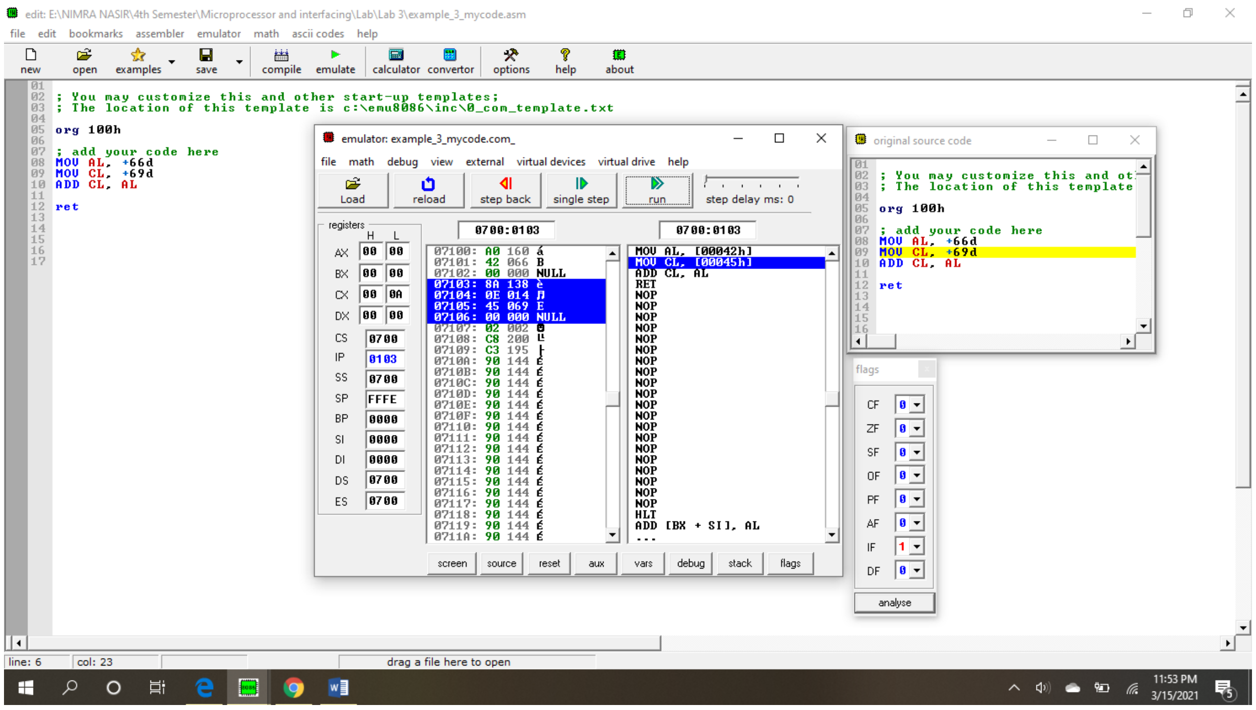
ADD CL, AL

Answer:

Since +66d = 66d as it doesn't effect the result in any way so we can consider it as 66d as well as same rule for the value +69d=69d then the code becomes: MOV AL, 66d MOV CL, 69d ADD CL, AL

Step#1:

This command is executed 66d is moved to register AL. After first step all the Flags are zero.



edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_3_mycode.asm

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV AL, +66d
MOV CL, +69d
ADD CL, AL

ret
```

emulator: example_3_mycode.com_

step delay ms: 0

registers

	H	L
AX	00	00
BX	00	00
CX	00	0A
DX	00	00
CS	0700	
IP	0103	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```
0700:0103

07100: A0 160 á
07101: 42 066 B
07102: 00 000 NULL
07103: 8A 138 è
07104: 0E 014 ♫
07105: 45 069 E
07106: 00 000 NULL
07107: 02 002 ☻
07108: C8 200 ╚
07109: C3 195 ├
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

```
0700:0103

MOV AL, [00042h]
MOV CL, [00045h]
ADD CL, AL
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
...
```

flags

CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

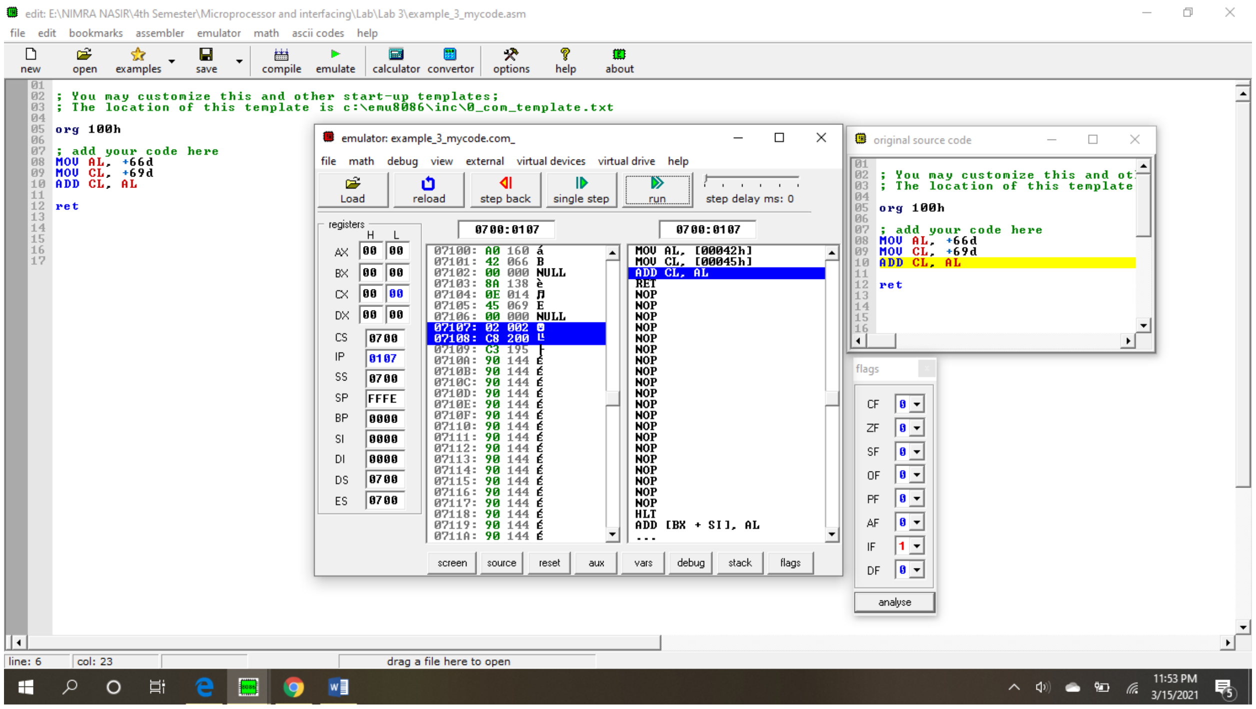
line: 6 col: 23 drag a file here to open

11:53 PM 3/15/2021

file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
Load reload step back single step run
screen source reset aux vars debug stack flags
analyse
original source code

Step#2:

This command is executed 69d is moved to register CL. After first step all the Flags are zero.



edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_3_mycode.asm

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV AL, +66d
MOV CL, +69d
ADD CL, AL

ret
```

emulator: example_3_mycode.com_

step delay ms: 0

registers

	H	L
AX	00	00
BX	00	00
CX	00	00
DX	00	00
CS	0700	
IP	0107	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```
0700:0107

07100: A0 160 á
07101: 42 066 B
07102: 00 000 NULL
07103: 8A 138 è
07104: 0E 014 ♫
07105: 45 069 E
07106: 00 000 NULL
07107: 02 002 ☻
07108: C8 200 ╚
07109: C3 195 ├
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

```
0700:0107

MOV AL, [00042h]
MOV CL, [00045h]
ADD CL, AL
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
...
```

flags

CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

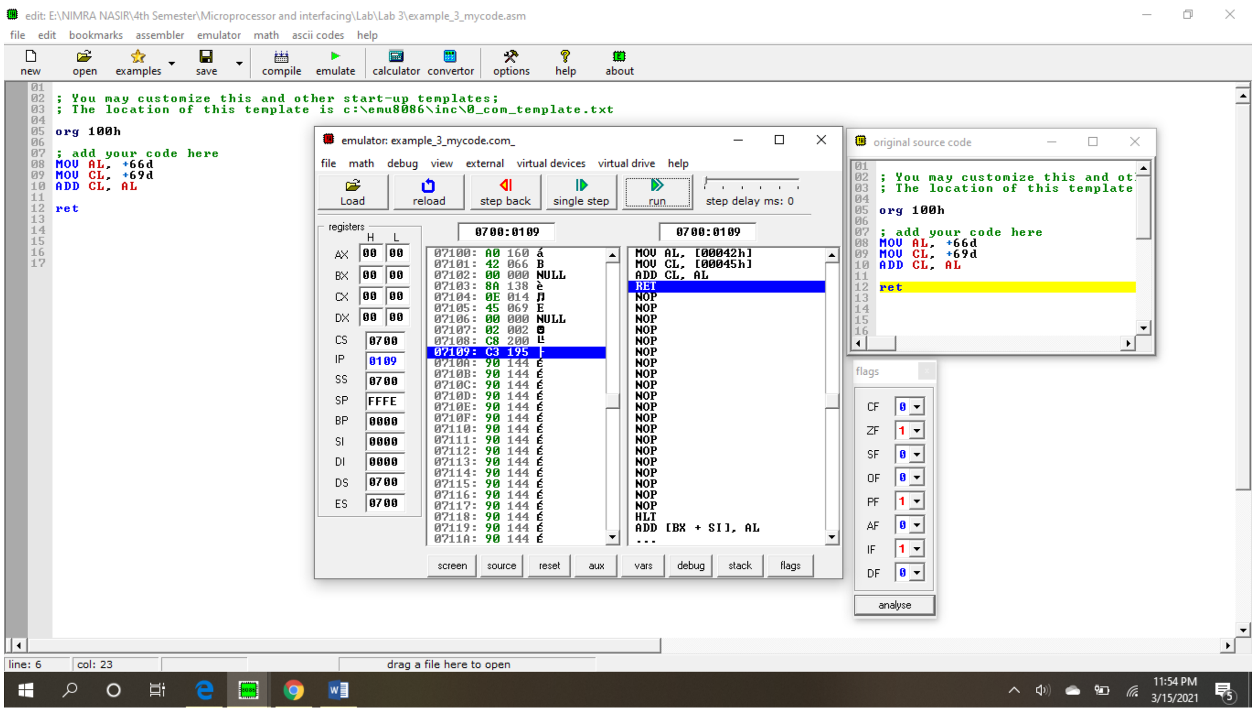
line: 6 col: 23 drag a file here to open

11:53 PM 3/15/2021

file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
Load reload step back single step run
screen source reset aux vars debug stack flags
analyse
original source code

Step#3:

The value AL will be added to the value of CL and as shown below in figure the parity bit is set to 1 because the system is even no one bit and as well as the OF and Sf also set to 1


edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_3_mycode.asm

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV AL, +66d
MOV CL, +69d
ADD CL, AL

ret
```

emulator: example_3_mycode.com_

registers

	H	L
AX	00	00
BX	00	00
CX	00	00
DX	00	00
CS	0700	
IP	0109	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```
0700:0109
07100: A0 160 á
07101: 42 066 B
07102: 00 000 NULL
07103: 8A 138 è
07104: 0E 014 ♫
07105: 45 069 E
07106: 00 000 NULL
07107: 02 002 ☻
07108: C8 200 ╚
07109: C3 195 ├
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

```
0700:0109
MOV AL, [00042h]
MOV CL, [00045h]
ADD CL, AL
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
...
```

original source code

```
; You may customize this and ot
; The location of this template

org 100h

; add your code here
MOV AL, +66d
MOV CL, +69d
ADD CL, AL

ret
```

flags

CF	0
ZF	1
SF	0
OF	0
PF	1
AF	0
IF	1
DF	0

line: 6 col: 23 drag a file here to open

11:54 PM 3/15/2021

Example: 4

MOV AL, -12

MOV BL, +18

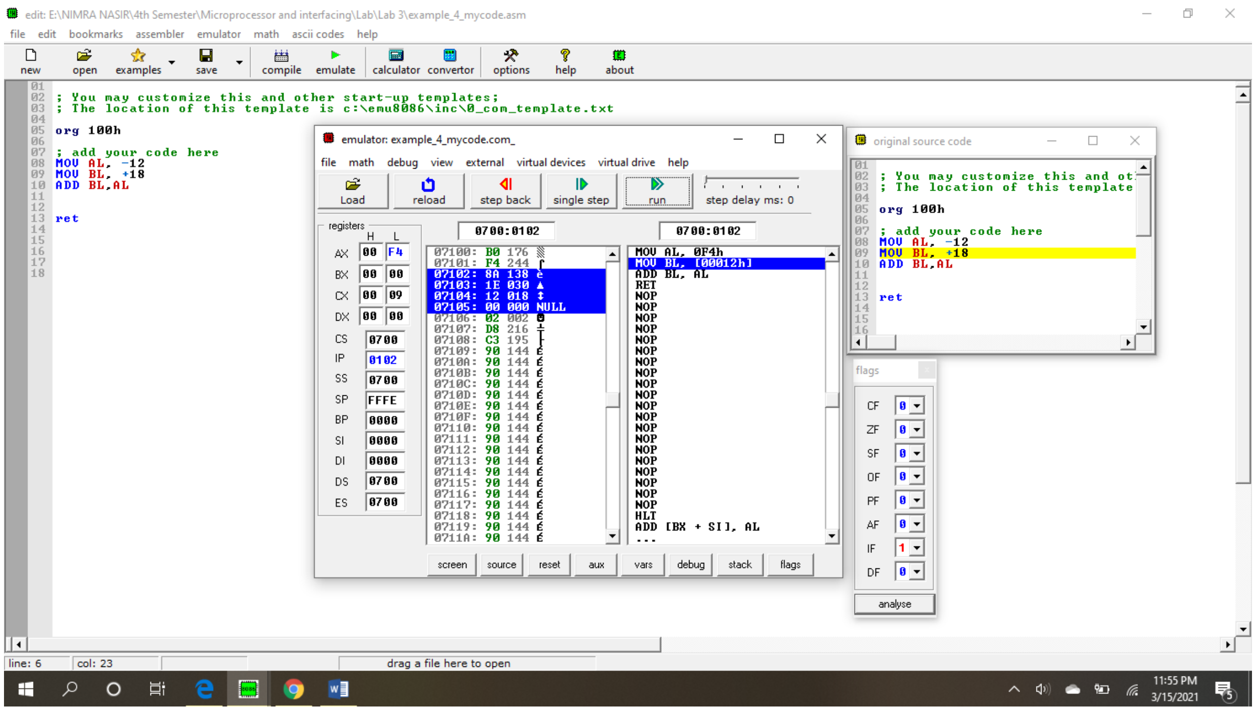
ADD BL, AL

Answer:

In the first step I have converted the signed decimal value -12 into the binary which is 11110100 And also considered the unsigned +18 decimal value as binary value which is 00010010 So the above code becomes: MOV AL, 11110100b MOV BL, 00010010b ADD BL,AL

Step#1:

In the step the value 11110100b will be moved in AL part of AX register and all the flags set to zero as shown.


edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_4_mycode.asm

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV AL, -12
MOV BL, +18
ADD BL,AL


ret
```

emulator: example_4_mycode.com_

registers	H	L
AX	00	F4
BX	00	00
CX	00	09
DX	00	00

CS	0700
IP	0102
SS	0700
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

```
0700:0102
07100: B0 176 ░
07101: F4 244 ⌠
07102: 8A 138 è
07103: 1E 030 ▲
07104: 12 018 ↕
07105: 00 000 NULL
07106: 02 002 ☻
07107: D8 216 ╪
07108: C3 195 ├
07109: 90 144 É
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

```
0700:0102
MOV AL, 0F4h
MOV BL, [00012h]
ADD BL, AL
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
...
```

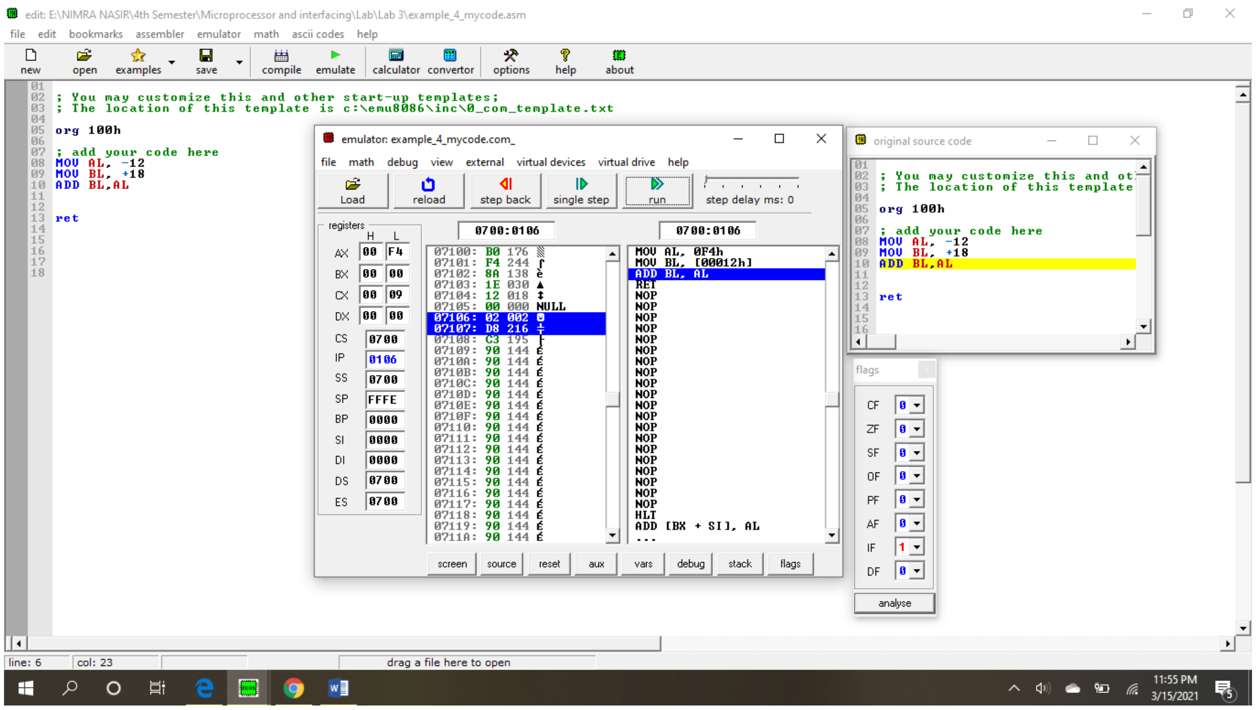
flags	
CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

line: 6 col: 23

file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags
original source code
analyse
drag a file here to open
11:55 PM
3/15/2021

Step#2:

Now the value 00010010b loaded to the Register's BX, BL part and all the falgs have zero bit value as shown.


edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_4_mycode.asm

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV AL, -12
MOV BL, +18
ADD BL,AL


ret
```

emulator: example_4_mycode.com_

registers	H	L
AX	00	F4
BX	00	00
CX	00	09
DX	00	00

CS	0700
IP	0106
SS	0700
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

```
0700:0106
07100: B0 176 ░
07101: F4 244 ⌠
07102: 8A 138 è
07103: 1E 030 ▲
07104: 12 018 ↕
07105: 00 000 NULL
07106: 02 002 ☻
07107: D8 216 ╪
07108: C3 195 ├
07109: 90 144 É
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

```
0700:0106
MOV AL, 0F4h
MOV BL, [00012h]
ADD BL, AL
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
...
```

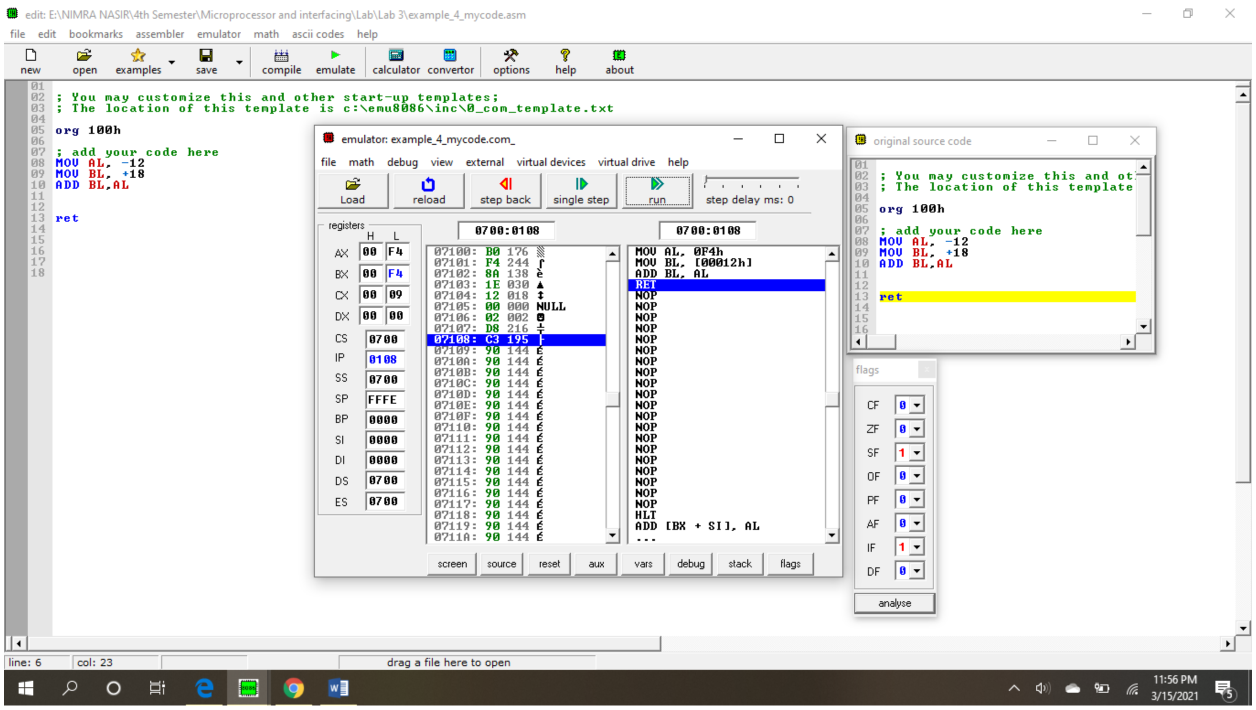
flags	
CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

line: 6 col: 23

file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags
original source code
analyse
drag a file here to open
11:55 PM
3/15/2021

Step#3:

In this step the BL AND AL registers value will be added and as we can see that the carry flag is set to 1 because it has the carry value 1 after the addition and we can also see that . Parity Flag (PF) is set to 1 because the system is even number one bit as shown in the following figure:



edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_4_mycode.asm

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV AL, -12
MOV BL, +18
ADD BL,AL

ret
```

emulator: example_4_mycode.com_

registers:

	H	L
AX	00	F4
BX	00	F4
CX	00	09
DX	00	00

CS	0700
IP	0108
SS	0700
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

```
0700:0108
07100: B0 176 ░
07101: F4 244 ⌠
07102: 8A 138 è
07103: 1E 030 ▲
07104: 12 018 ↕
07105: 00 000 NULL
07106: 02 002 ☻
07107: D8 216 ╪
07108: C3 195 ├
07109: 90 144 É
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

```
0700:0108
MOV AL, 0F4h
MOV BL, [00012h]
ADD BL, AL
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
...
```

original source code

```
; You may customize this and ot
; The location of this template

org 100h

; add your code here
MOV AL, -12
MOV BL, +18
ADD BL,AL

ret
```

flags:

CF	0
ZF	0
SF	1
OF	0
PF	0
AF	0
IF	1
DF	0

file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags
analyse
line: 6 col: 23 drag a file here to open
11:56 PM 3/15/2021

<u>Example: 5</u>

MOV AH, -30

MOV DL, +14

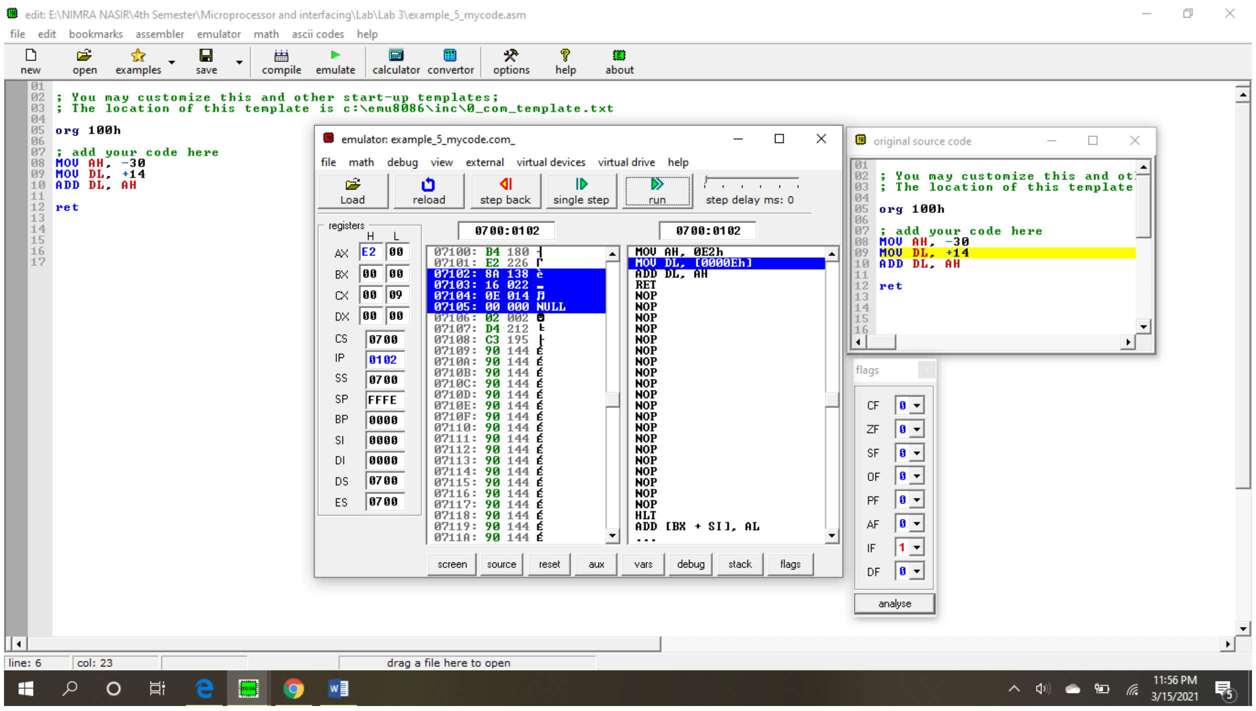
ADD DL, AH

<u>Answer:</u>

First I have considered the signed -30 decimal value as 8 bit binary value which is 11100010 and as well as the unsigned +14 decimal value as 8 bit binary value which becomes 00001110 then the code becomes: MOV AH, 11100010b MOV DL, 00001110b ADD DL, AH

<u>Step#1:</u>

In this step the 8 bit binary value 11100010 will be inserted into the AH part of AX register and as shown in the following figure the flags have zero value.



edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_5_mycode.asm

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV AH, -30
MOV DL, +14
ADD DL, AH

ret
```

emulator: example_5_mycode.com_

	H	L
AX	E2	00
BX	00	00
CX	00	09
DX	00	00
CS	0700	
IP	0102	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```
0700:0102
07100: B4 180 ┤
07101: E2 226 Γ
07102: 8A 138 è
07103: 16 022 ▬
07104: 0E 014 ♫
07105: 00 000 NULL
07106: 02 002 ☻
07107: D4 212 ╘
07108: C3 195 ├
07109: 90 144 É
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

```
0700:0102
MOV AH, 0E2h
MOV DL, [0000Eh]
ADD DL, AH
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
...
```

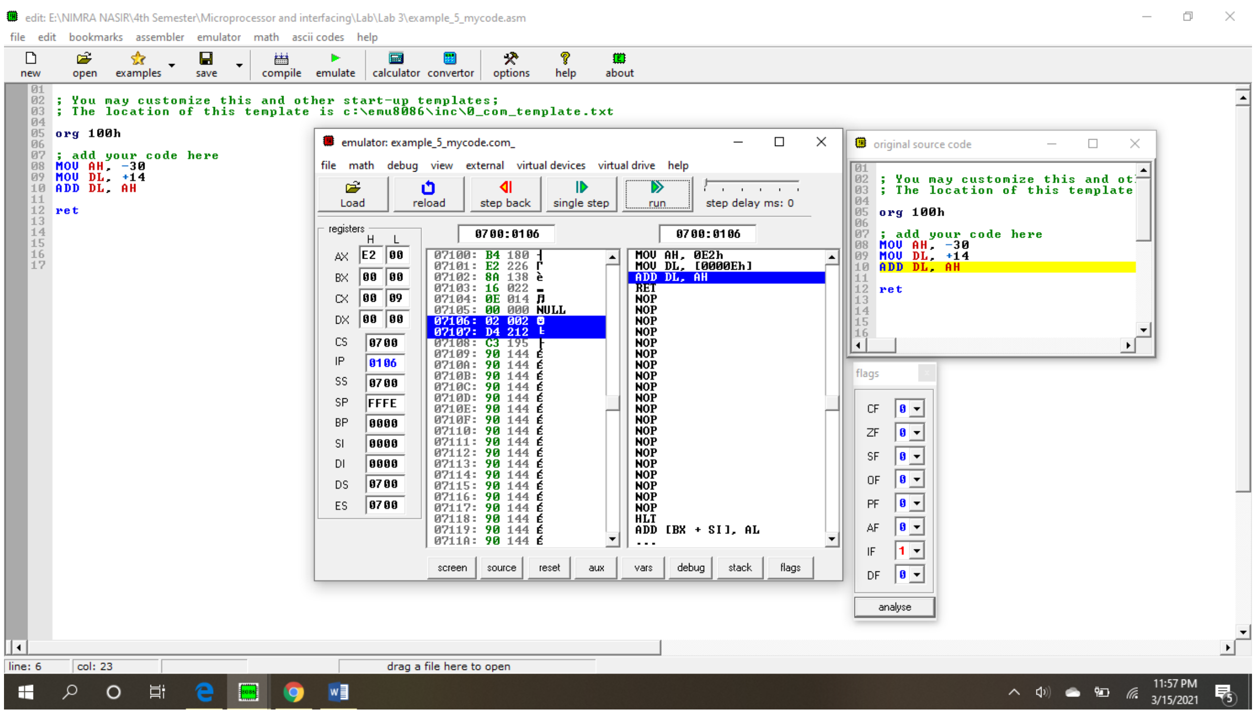
flags	
CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

line: 6 col: 23 drag a file here to open

11:56 PM 3/15/2021

Step#2:

Now the value 00001110 will be loaded to the DL register and all the flags have zero value as shown bellow:


edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_5_mycode.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV AH, -30
MOV DL, +14
ADD DL, AH

ret
```

emulator: example_5_mycode.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers

	H	L
AX	E2	00
BX	00	00
CX	00	09
DX	00	00

CS	0700
IP	0106
SS	0700
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

```
0700:0106
07100: B4 180 ┤
07101: E2 226 Γ
07102: 8A 138 è
07103: 16 022 ▬
07104: 0E 014 ♫
07105: 00 000 NULL
07106: 02 002 ☻
07107: D4 212 ╘
07108: C3 195 ├
07109: 90 144 É
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

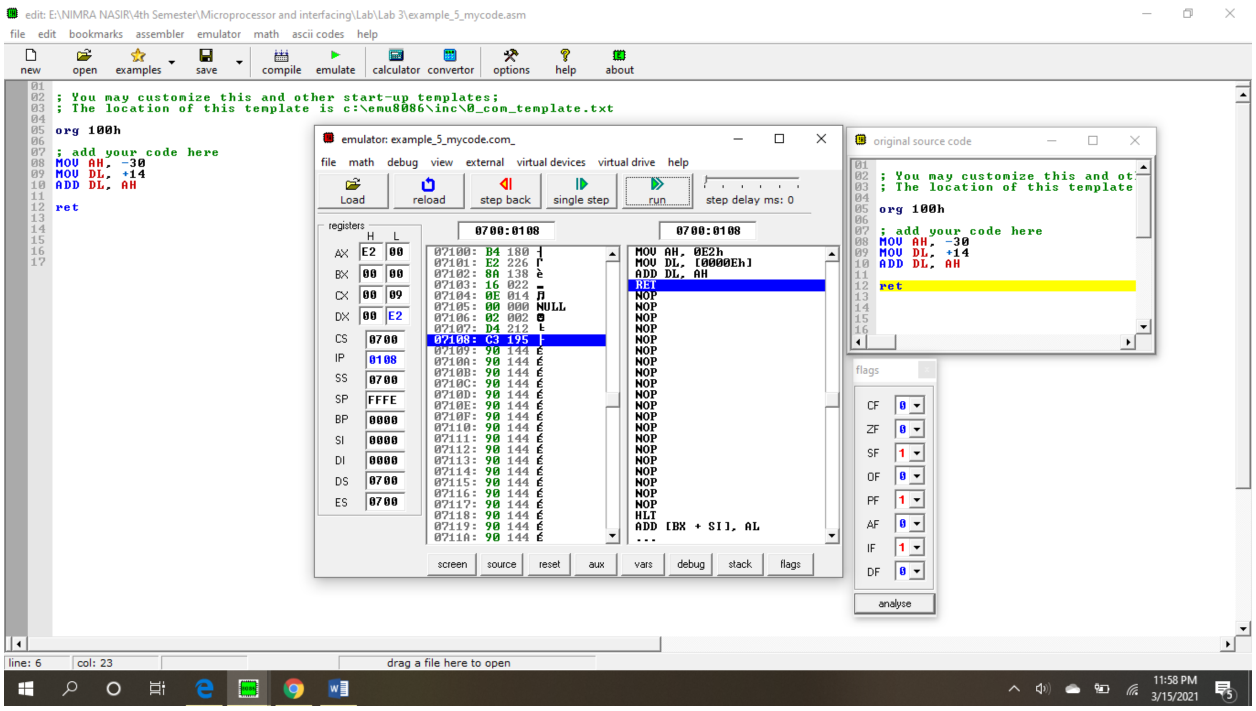
```
0700:0106
MOV AH, 0E2h
MOV DL, [0000Eh]
ADD DL, AH
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
...
```

```
; You may customize this and ot
; The location of this template

org 100h

; add your code here
MOV AH, -30
MOV DL, +14
ADD DL, AH

ret
```

CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

Step#3:

In this step the DL AND AH value will be added and result will be stored in the DL register and Parity Flag (PF) is set to 1 because the system is even number one bit as shown below and auxiliary flag is also set to 1 as shown in following figure because it Set to 1 when there is an unsigned overflow for low nibble (4 bits) and we can also see that sign flag is also active now as it set to 1 when result is negative. When result is positive it is set to 0. Actually this flag take the value of the most significant bit. So the result is negative in that case.



edit: E:\NIMRA NASIR\4th Semester\Microprocessor and interfacing\Lab\Lab 3\example_5_mycode.asm

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

; add your code here
MOV AH, -30
MOV DL, +14
ADD DL, AH

ret
```

line: 6 col: 23

emulator: example_5_mycode.com_

```
registers
      H   L
AX   E2  00
BX   00  00
CX   00  09
DX   00  E2
CS   0700
IP   0108
SS   0700
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
0700:0108

07100: B4 180 ┤
07101: E2 226 Γ
07102: 8A 138 è
07103: 16 022 ▬
07104: 0E 014 ♫
07105: 00 000 NULL
07106: 02 002 ☻
07107: D4 212 ╘
07108: C3 195 ├
07109: 90 144 É
0710A: 90 144 É
0710B: 90 144 É
0710C: 90 144 É
0710D: 90 144 É
0710E: 90 144 É
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
```

```
0700:0108

MOV AH, 0E2h
MOV DL, [0000Eh]
ADD DL, AH
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
...
```

```
flags
CF 0
ZF 0
SF 1
OF 0
PF 1
AF 0
IF 1
DF 0
```

file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags
original source code
analyse
drag a file here to open
11:58 PM
3/15/2021

--THE END--