

# LAB REPORT FOR EXP 2

COURSE TITLE : EEE 416

Name : Md Maisoon Rahman  
ID : 1606038  
Sec : A2

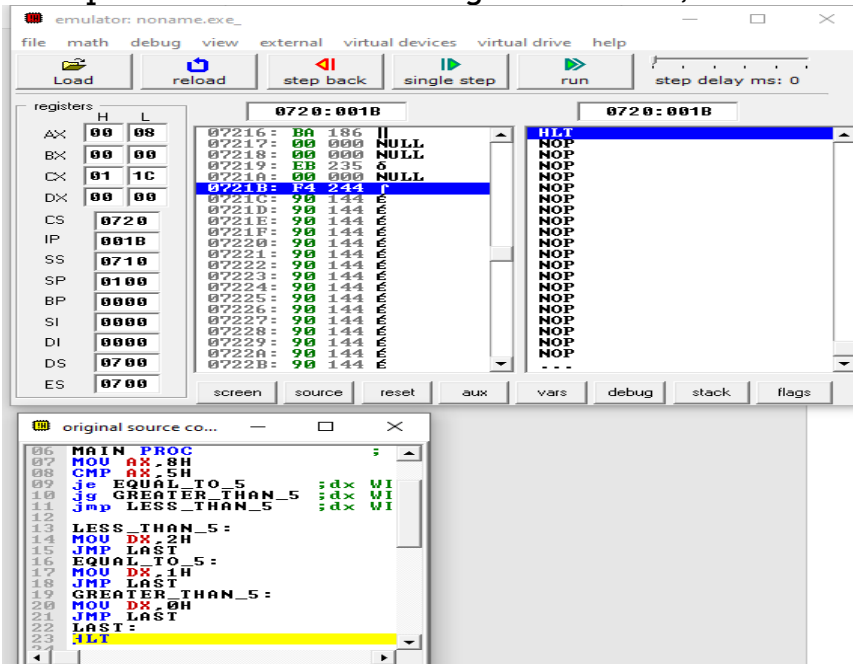
## HOMEWORK REPORT 1

### CODE :

```
01 .MODEL SMALL ;depends on the size of the code
02 .STACK 100H ; 100h hexadecimal stack memory
03 .DATA
04
05 .CODE
06 MAIN PROC ; almost like int main
07     MOV AX,3H
08     CMP AX,5H
09     je EQUAL_TO_5 ;dx WILL BE 1
10     jg GREATER_THAN_5 ;dx WILL BE 0
11     jmp LESS_THAN_5 ;dx WILL BE 2
12
13 LESS_THAN_5:
14     MOV DX,2H
15     JMP LAST
16 EQUAL_TO_5:
17     MOV DX,1H
18     JMP LAST
19 GREATER_THAN_5:
20     MOV DX,0H
21     JMP LAST
22 LAST:
23     HLT
24
25     MAIN ENDP
26 END MAIN ;
```

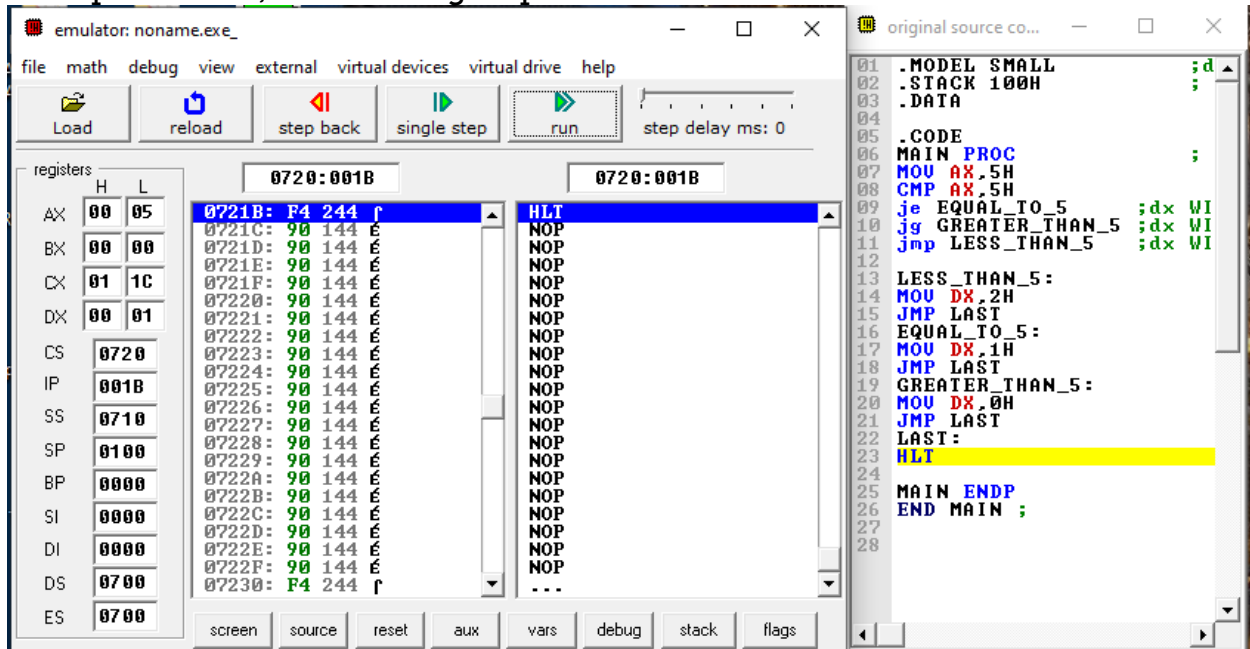
### OUTPUT :

For input case 8h which will be greater than 5h, the following result was obtained



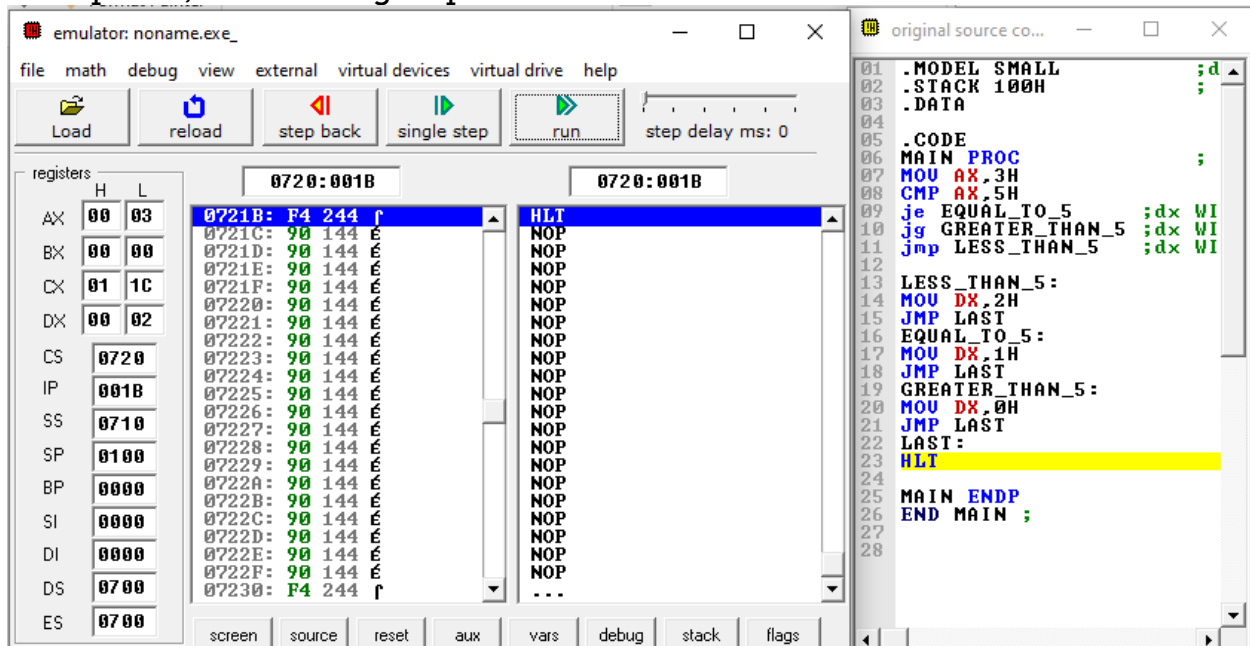
Here, as required, 0 is assigned to DX, as the number is greater than 5h.

For input case 5h, the following output was obtained:



Here, as required, 1 is assigned to DX, as the number is equal to 5h

For input 3h, the following output was obtained:



Here, as required, 2 is assigned to DX, as the number is less than 5h

## HOMEWORK REPORT 2

### CODE:

```
01 CODE SEGMENT
02 ASSUME CS:CODE, DS:CODE
03
04 MOV AX,86B1H
05 MOV BX,3F42H
06
07 SUB AX,BX
08 JO OVERFLOW_HOISE
09 JNO OVERFLOW_HOYNAI
10 HLT
11
12 OVERFLOW_HOISE:
13 MOV CX,0
14 JMP LAST
15 OVERFLOW_HOYNAI:
16 MOV CX,1
17 JMP LAST
18
19 LAST:
20 HLT
21 CODE ENDS
22 END
```

### OUTPUT:

For 86B1 - 3F42 , there will be overflow.

Therefore, for overflow, register CX should be 0. The desired output was observed in our simulation.

The screenshot displays an x86 emulator interface. On the left, the 'registers' window shows the state of various registers: AX (47:6F), BX (3F:42), CX (00:00), DX (00:00), CS (0100), IP (0017), SS (0100), SP (FFFE), BP (0000), SI (0000), DI (0000), DS (0100), and ES (0100). The central instruction window shows the current instruction is 'HLT' at address '01017'. The right-hand window displays the assembly source code, with the 'HLT' instruction at line 10 highlighted in yellow, indicating the current point of execution.

## HOMEWORK REPORT 3

### CODE:

```
01 CODE SEGMENT
02 ASSUME CS:CODE, DS:CODE
03
04 MOV AX,10H      ;INPUT X
05 MOV BX,11H      ;INPUT Y
06
07 CMP AX,1000H
08 JG SUMMATION    ;IS X>1000?
09
10 CMP BX,1000H
11 JNG SUBTRACTION;IS Y<1000?
12
13 SUMMATION:
14 CMP BX,100H
15 JL BOTH_OK      ; x>1000 AND Y<100
16 ADD AX,BX
17 JMP LAST
18
19 SUBTRACTION:
20 SUB AX,BX
21 JMP LAST
22
23 BOTH_OK:
24 NOT AX
25 JMP LAST
26
27 LAST:
28 HLT
29 CODE ENDS
30 END
```

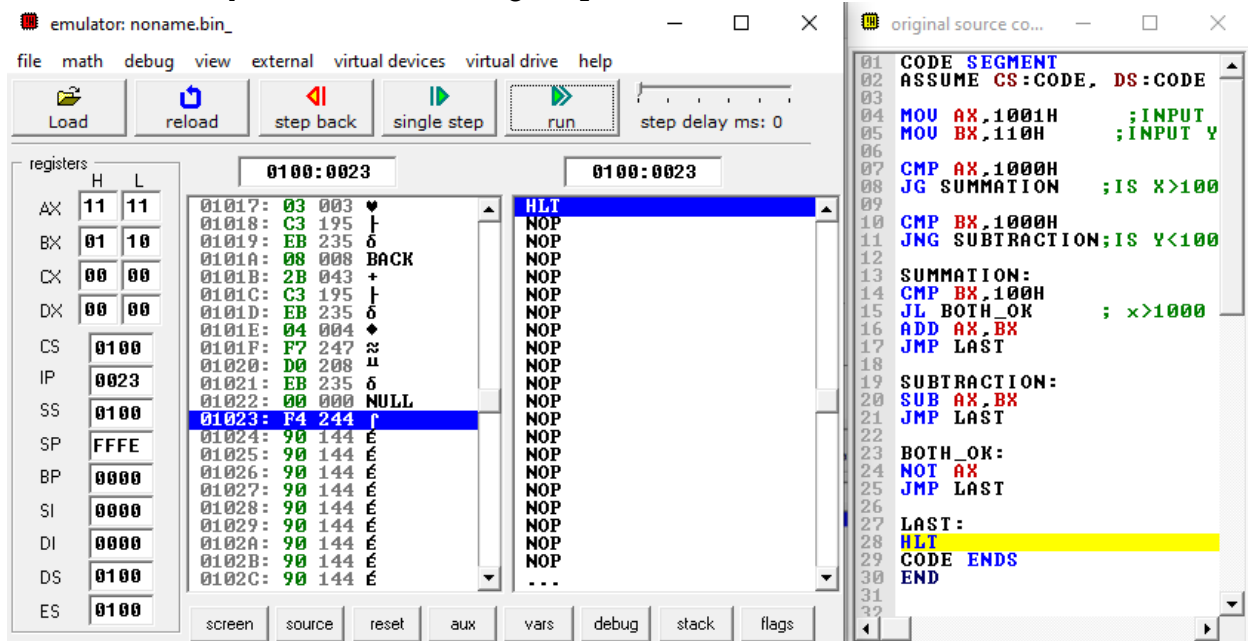
### OUTPUT:

For output x=1001 and y=11;

as x>1000 and y<100 ; the x=x' command is carried out

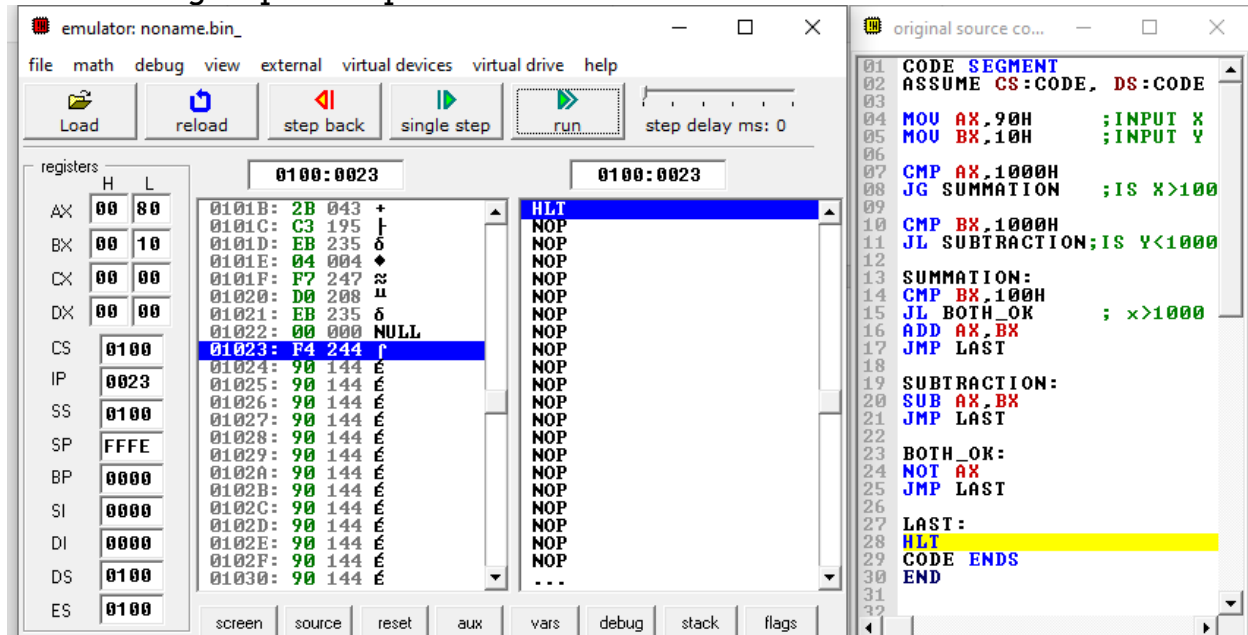
The screenshot displays an x86 emulator interface. The top menu bar includes 'file', 'math', 'debug', 'view', 'external', 'virtual devices', 'virtual drive', and 'help'. Below the menu is a toolbar with buttons for 'Load', 'reload', 'step back', 'single step', 'run', and a 'step delay ms: 0' slider. The main window is divided into several panes. On the left, the 'registers' pane shows the state of various registers: AX (EF, FE), BX (00, 11), CX (00, 00), DX (00, 00), CS (0100), IP (0023), SS (0100), SP (FFFE), BP (0000), SI (0000), DI (0000), DS (0100), and ES (0100). The central pane shows memory addresses and their contents, with the instruction at address 01023 (HLT) highlighted. The right pane shows the source code with the instruction 'JNG SUBTRACTION;IS Y<100' highlighted. The bottom of the window has tabs for 'screen', 'source', 'reset', 'aux', 'vars', 'debug', 'stack', and 'flags'.

For x=1000 and y=110;  
as x>1000 and y>100, the following output was obtained.



As expected, x+y command is carried out.

For x=90 , y=80 ; the following command should have been carried out  
x=x-y  
The following output is expected : x = 10



## LEAP YEAR

### CODE:

```
01 CODE SEGMENT
02
03 ASSUME CS:CODE, DS:CODE
04
05 MOV AX,CS ; these two lines are here to ensure DS=CS. Remember, MOV DS,CS is illegal
06
07 MOV DS,AX ; you may ignore these line in EMU. But they are necessary in Hardware
08
09
10
11 MOV AX, YEAR
12
13 MOV DX,0 ; be careful to put 0 in DX before you use DIU in case of 16 bit operand
14
15 ; As DX:AX is dividend, if there is garbage value in DX, you'll get ; specious result
16 ;WRITE YOUR CODE HERE
17 MOV BX,4d
18 DIV BX ;IF LEAP YEAR, REMAINDER DX =0
19 CMP DX,0d
20 JNE NOT_LEAP_YEAR
21
22 ;DIVISIBILITY BY 100
23 MOV AX, YEAR
24 MOV DX,0d
25 MOV BX,100d
26
27 DIV BX
28 CMP DX,0d
29 JNE LEAP_YEAR
30
31 MOV AX, YEAR
32 MOV DX,0d
33 MOV BX,400d
34 ;DIVISIBILITY BY 400
35 DIV BX
36 CMP DX,0d
37 JE LEAP_YEAR
38 JNE NOT_LEAP_YEAR
39
40 LEAP_YEAR:
41 MOV LEAPYEAR,1
42 HLT
43
44 NOT_LEAP_YEAR:
45 MOV LEAPYEAR,0
46 HLT
47
48
49 HLT
50
51
52
53 YEAR DW 1900d
54
55 LEAPYEAR DB ?
56
57 CODE ENDS
58
59 END
```

### OUTPUT:

For Year=2021

The screenshot displays an 8086 emulator interface with three main panels:

- Registers:** Shows the state of the 8086 registers. DX:AX is 0000:0021, indicating the dividend is 21 (the year 2021). Other registers like BX, CX, SI, DI, etc., contain various values.
- Source:** Displays the assembly code being executed. The instruction at address 0100:003F is `HLT`, which is highlighted in blue. The code includes comments and instructions for checking divisibility by 100 and 400.
- Variables:** Shows the values of variables. `YEAR` is 07E5h (1900) and `LEAPYEAR` is 00h (0).

The status bar at the bottom indicates the current step is 0100:003F.

The screenshot displays the x86-64 emulator interface with the following components:

- Emulator Title Bar:** emulator: noname\_bin\_
- Menu Bar:** file, math, debug, view, external, virtual devices, virtual drive, help
- Toolbar:** Load, reload, step back, single step, run, step delay ms: 0
- Registers Panel:**
  - H (High) L (Low):**
    - AX: 00 04
    - CX: 01 90
    - DX: 00 00
    - DX: 01 2C
    - CS: 0100
    - IP: 0041
    - SS: 0100
    - SP: FFFE
    - BP: 0000
    - SI: 0000
    - DI: 0000
    - DS: 0100
    - ES: 0100
- Memory Panel:**
  - Address: 0100:0041
  - Content: 0102D: F7 247 2, 0102E: F3 243 1, 0102F: 83 131 1, 01030: FA 250 1, 01031: 00 000 NULL, 01032: 74 116 t, 01033: 02 002 t, 01034: 75 117 u, 01035: 06 006 NOP, 01036: C6 198 t, 01037: 06 006 t, 01038: 45 069 E, 01039: 00 000 NULL, 0103A: 01 001 t, 0103B: F4 244 f, 0103C: C6 198 t, 0103D: 06 006 t, 0103E: 45 069 E, 0103F: 00 000 NULL, 01040: 00 000 NULL, 01041: F4 244 f, 01042: F4 244 f
- Assembly Panel:**
  - Address: 0100:0041
  - Code:
 

```

16 ;WRITE YOUR CODE HERE
17 MOV BX,4d
18 DIV BX ;IF LEAP_YEAR
19 CMP DX,0d
20 JNE NOT_LEAP_YEAR
21
22 ;DIVISIBILITY BY 100
23 MOV DX,0d
24
25 MOV BX,100d
26
27 DIV BX
28 CMP DX,0d
29 JNE LEAP_YEAR
30
31 MOV AX, YEAR
32 MOV DX,0d
33 MOV BX,400d
34 ;DIVISIBILITY BY 400
35 DIV BX
36 CMP DX,0d
37 JE LEAP_YEAR
38 JNE NOT_LEAP_YEAR
39
40 LEAP_YEAR:
41 MOV LEAPYEAR,1
42 HLT
43
44 NOT_LEAP_YEAR:
45 MOV LEAPYEAR,0
46 HLT
          
```
- Variables Panel:**
  - Size: byte, Elements: 1
  - Variable: YEAR, Address: 076Ch, Value: LEAPYEAR 00h
- Footer:** screen, source, reset, aux, vars, debug, stack, flags

The screenshot displays the DOSBox emulator interface. The top menu bar includes 'file', 'math', 'debug', 'view', 'external', 'virtual devices', 'virtual drive', and 'help'. Below the menu is a toolbar with icons for 'Load', 'reload', 'step back', 'single step', 'run', and a 'step delay ms: 0' input field.

The 'registers' panel on the left shows the state of various registers:

	H	L
AX	00	14
CX	00	64
BX	00	00
DX	00	14
CS	0100	
IP	003B	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

The main window is split into two panes. The left pane shows a memory dump for address 0100:003B, with the selected address 0103B containing the value F4 244. The right pane displays the assembly code, with the instruction 'HLT' at address 0103B highlighted. The assembly code includes comments and instructions for a program that checks for a leap year.

On the right side of the interface, there is a 'variables' panel. It shows a list of variables with their values and data types. The 'LEAPYEAR' variable is highlighted, showing a value of 01h and a data type of 'byte'.

The screenshot displays the x86-64 emulator interface with the following components:

- Top Bar:**
  - Emulator: noname\_bin.exe
  - File menu: file, math, debug, view, external, virtual devices, virtual drive, help
  - Buttons: Load, reload, step back, single step, run, step delay ms: 0
- Registers Panel:**

	H	L
AX	00	05
BX	01	90
CX	00	00
DX	00	00
CS	0100	
IP	003B	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	
- Memory Panel:**

Address	Hex	ASCII
0100D: F7	247	≈
0102E: F3	243	≈
0102F: 83	131	â
01030: F8	250	•
01031: 00	000	NULL
01032: 74	116	•
01033: 02	002	•
01034: 75	117	•
01035: 06	006	•
01036: C6	198	•
01037: 06	006	•
01038: 45	069	E
01039: 00	000	NULL
0103A: 01	001	•
0103B: F4	244	•
0103C: C6	198	•
0103D: 06	006	•
0103E: 45	069	E
0103F: 00	000	NULL
01040: 00	000	NULL
01041: F4	244	•
01042: F4	244	•
- Assembly Panel:**

```

12 MOV DX,0 ; be careful to
13
14 ; As DX:AX is dividend,
15 ;WRITE YOUR CODE HERE
16 MOV BX,4d
17 DIV BX ;IF LEAP YEAR
18 CMP DX,0d
19 JNE NOT_LEAP_YEAR
20
21 ;DIVISIBILITY BY 100
22 MOV AX, YEAR
23 DIV DX,0d
24 MOV BX,100d
25
26
27 DIV BX
28 CMP DX,0d
29 JNE LEAP_YEAR
30
31 MOV AX, YEAR
32 MOV DX,0d
33 MOV BX,400d
34 ;DIVISIBILITY BY 400
35 DIV BX
36 CMP DX,0d
37 JE LEAP_YEAR
38 JNE NOT_LEAP_YEAR
39
40 LEAP_YEAR:
41 MOV LEAPYEAR,1
42 HLT
  
```
- Variables Panel:**

Variable	Value
YEAR	07D0h
LEAPYEAR	01h