

Computer Organization and Architecture (EET2211)

LAB II: Analyze and Evaluate the Branching operation in the 8086 Microprocessor.

Siksha 'O' Anusandhan (Deemed to be University),
Bhubaneswar

Branch: CSE		Section: 44	
S. No.	Name	Registration No.	Signature
27	E. Jagadeeswar Patro	2241016309	E. Jagadeeswar Patro

Marks: ____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1. Find the sum and average of N 16-bit numbers.
2. Count no. of 0's in an 8-bit number.
3. Move a block of 16-bit data from one location to other.
4. Multiplication of two 16-bit numbers without using MUL instruction in direct addressing mode.

II. PRE-LAB

Note: For each objective in prelab describe the following points:

- Write the pseudocode.
- Write the assembly code with description (ex. Mov ax,3000h - ax<-3000h)
- Examine & analyze the input/output of assembly code.

Objective 1:-

pseudocode:

set SI to 2000h

load the byte at address SI into CL

Set CH to 00H

Copy CX into BX

Set AX to 0000H

Loop1: Increment SI

Increment SI

Add the data at address SI to AX

Jump if no carry to Loop2

Increment CH

Loop2: Decrement CL

Jump if not zero to loop1

Increment SI

Increment SI

Store AX at address SI

Increment SI

Increment SI

Store CH at address SI

Move CH into DL

Divide DX:AX by BX

Increment SI

Increment SI

Store AX at address SI

Increment SI
Increment SI
Store DX at address SI
Halt

Assembly Code :

```
mov ax, 0000h  
mov ds, ax  
mov SI, 2000h  
mov cl, [SI]  
mov ch, 00h  
mov bx, cx  
mov ax, 0000h
```

```
loop1: inc SI  
       inc SI  
       add ax, [SI]  
       jnc loop2  
       inc ch
```

```
loop2: dec ch  
       jnz loop1  
       inc SI  
       inc SI  
       mov [SI], ax  
       inc SI  
       inc SI  
       mov [SI], ch  
       mov dl, ch  
       div bx  
       inc SI  
       inc SI  
       mov [SI], ax  
       inc SI  
       inc SI  
       mov [SI], dx  
hlt
```

Objective 2 : Count no of 0's in an 8-bit number
pseudocode:

input 8-bit data at [2000h]

store it at al

cl = 08h

loop 1: shift al right by 1 bit

Jump if carry to loop2

Increment CH

loop 2: Decrement CL

Jump if not zero to loop1

store value of ch at [2001h]

hlt

Assembly Code :

mov ax, 0000h

mov ds, ax

mov al, [2000h]

mov cl, 08h

loop 1: shr al, 01

jc loop2

inc ch

loop 2: dec cl

jnz loop1

mov [2001h], ch

hlt

Objective 3: Move a block of 16-bit data from one location to another.

pseudocode:

set SI to 2000h for input
set DI to 2010h for output
set CL to 05h ; count value

loop: move the data at address SI to BX

store BX at address DI

Increment SI 2 times

Increment DI 2 times

Decrement CL

jump if not zero to 'loop'

h-alt

Assembly Code:

mov ax, 0000h

mov ds, ax

mov SI, 2000h

mov DI, 2010h

mov cl, 05

loop: mov bx, [SI]

mov [DI], bx

inc SI

inc SI

inc DI

inc DI

dec cl

jnz loop

hlt

Objective 4: Multiplication of two 16-bit numbers without using mul instruction in direct addressing mode.

pseudocode -

move data at address 20001H to BX

move data at address 20002H to CX

Set DX to 0000H ; count is initialized to zero

Loop 1:

AX + BX stored at AX

Jump if no carry to Loop 2

Increment DX

Loop 2:

Decrement CX

Jump if not zero to Loop 1

store AX at address 2010h

store DX at address 2012h

halt

Assembly Code :-

mov ax, 0000h

mov ds, ax

mov bx, [2000h]

mov cx, [2002h]

mov dx, 0000h

loop1: add ax, bx

jnc loop2

inc dx

loop2: dec cx

jnz loop1

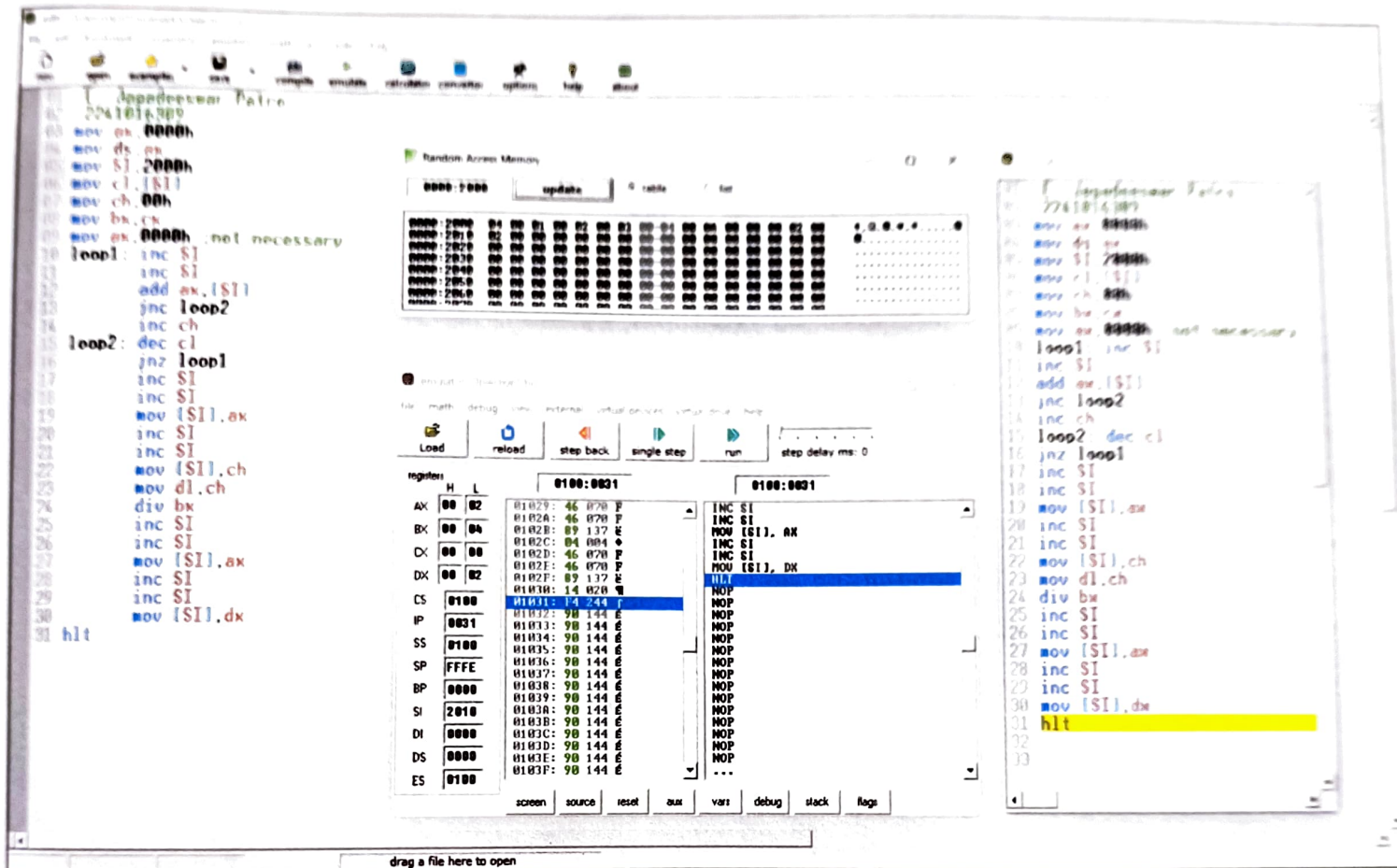
mov [2010h], ax

mov [2012h], dx

hlt

III. LAB

Objective 1:



From this result, I have observed.....

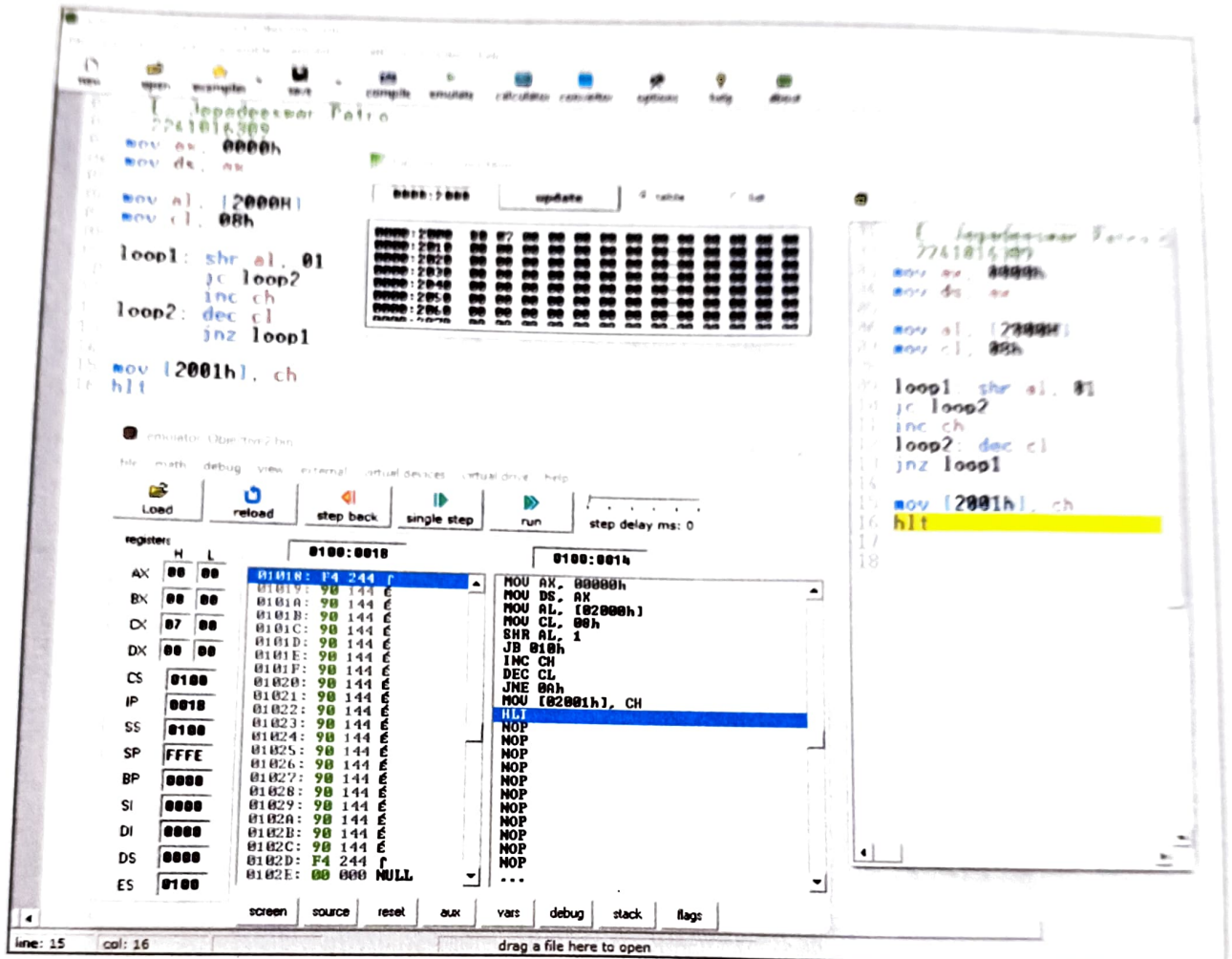
Input:

Sl. No.	Memory Location	Operand (Data)
1	0000:2000	04
2	0000:2002	01
3	0000:2004	02
4	0000:2006	03
5	0000:2008	04

Output:

Sl. No.	Memory Location	Operand (Data)
1	0000:200A	0A
2	0000:200E	02
3	0000:2010	02

Objective 2:



From this result, I have observed.....

Input:

Sl. No.	Memory Location	Operand (Data)
1	0000:2000	80

Output:

Sl. No.	Memory Location	Operand (Data)
1	0000:2001	07

Objective 3:



From this result, I have observed.....

Input:

Sl. No.	Memory Location	Operand (Data)
1	0000:2000	12 12
2	0000:2002	13 13
3	0000:2004	14 14
4	0000:2006	15 15
5	0000:2008	16 16

Output:

Sl. No.	Memory Location	Operand (Data)
1	0000:2010	12 12
2	0000:2012	13 13
3	0000:2014	14 14
4	0000:2016	15 15
5	0000:2018	16 16

Objective 4:

The screenshot displays a DOS emulator interface. On the left, the assembly code is shown with line numbers 01 through 17. The code includes instructions for moving values into registers, setting up loops, and halting. On the right, a 'Random Access Memory' window shows a memory dump starting at address 0000:2000. Below the code, the 'Registers' window shows the state of various registers, including AX, BX, CX, DX, CS, IP, SS, SP, BP, SI, DI, DS, and ES. The 'Source' window shows the assembly code with line 17 highlighted.

```

01 : E. Jagadeeswar Patro
02 : 2241016309
03
04 mov ax, 0000h
05 mov ds, ax; ds=0000h
06 mov bx, [2000h]
07 mov cx, [2002h]
08 mov dx, 0000h
09
10 loop1: add ax, bx
11         jnc loop2
12         inc dx
13 loop2: dec cx
14         jnz loop1
15         mov [2010h], ax
16         mov [2012h], dx
17 hlt
  
```

Random Access Memory

Address	0000:2000	0000:2010	0000:2020	0000:2030	0000:2040	0000:2050	0000:2060
0000:2000	00 00 01 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0000:2010	18 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0000:2020	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0000:2030	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0000:2040	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0000:2050	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0000:2060	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

Registers

Register	H	L
AX	00	18
BX	00	00
CX	00	00
DX	00	00
CS	0100	
IP	001F	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0000	
ES	0100	

Source

```

01 : E. Jagadeeswar Patro
02 : 2241016309
03
04 mov ax, 0000h
05 mov ds, ax; ds=0000h
06 mov bx, [2000h]
07 mov cx, [2002h]
08 mov dx, 0000h
09
10 loop1: add ax, bx
11         jnc loop2
12         inc dx
13 loop2: dec cx
14         jnz loop1
15         mov [2010h], ax
16         mov [2012h], dx
17 hlt
  
```

From this result, I have observed.....

Input:

Sl. No.	Memory Location	Operand (Data)
1	0000:2000	08
2	0000:2002	03

Output:

Sl. No.	Memory Location	Operand (Data)
1	0000:2010	18
2	0000:2012	00

IV. CONCLUSION

↳ In this experiment we learn how to perform various operations in the x86 assembly language, such as use of loops to perform a set of instructions repeating by checking the flags.

V. POST LAB

1. Analyze the following code and find out the value of registers.

```
MOV AX, 4246H → AX = 4246H
MOV BX, 123FH → BX = 123FH
AND AX, BX → AX AND BX = 022H
ADD AX, BX → AX + BX = (022H) + (123FH) = (1246H)
ROR AX, 02H → AX >> 2 = 4904H
INC BX → 123FH + 1 = 1240H
INC BX → 1240H + 1 = 1241H
MOV [BX], AX → Store at address of BX
HLT → Stop program
```

After Executing, AX = 4904H
BX = 1241H

2. Division of two 16-bit numbers without using DIV instruction in direct addressing mode.

```
MOV AX, [3000H]
MOV BX, [3002H]
XOR CX, CX
L1: CMP AX, BX
    JB L2
    SUB AX, BX
    INC CX
    JMP L1
L2: MOV [2000H], CX
    MOV [2002H], AX
HLT
```