Computer Organization and Architecture (EET2211)

LAB I: Analyze the Arithmetic and Logical operations using different Addressing Modes of the 8086 Microprocessor.

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

Branch: Computer Science & Engineering		Section: 201044	
S. No.	**Name**	**Registration No.**	**Signature**
27	E. Jagadeeswar Patro	2241016309	E. Jagadeeswar Patro

Marks: _____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1. Perform Addition, Subtraction, Multiplication, and Division of two 16-bit numbers using immediate addressing mode and store the results using direct addressing mode.
2. Perform the following operations on two 8-bit data (**data1**, **data2**) given in memory locations and store the result in another memory location using indirect addressing mode.
   i. Swapping of nibble of **data1**
   ii. Y= (**data1** and **data2**) or (**data1** xor **data2**)
3. Find the Gray code of an 8-bit binary number.
4. Find the 2's complement of an 8-bit number.

II. PRE-LAB

- Explain the addressing modes involved in instructions.

Immediate: data is a part of instruction & appears in the form of successive byte or bytes.

Direct: a 16-bit memory address (offset) is directly specified in the instruction.
PA = ds × 10h + 2000h

Indirect: The offset addressing data is in either BX or SI or DI registers.

For each objective in prelab describe the following points:

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

Obj 1:

```
mov au, 0000h
mov ds, au ; DS=0000H
mov au, 3457h ; Input 1
mov cu, au ; value of au stored in cu
mov bu, 0015h ; Input 2
add au, bu ; addition of au & bu & ans stored in au.
mov [2000h], au ; answer stored in [2000h] memory location.
mov au, cu ; getting original value of au
sub au, bu ; subtracting and storing in au.
mov [2010h], au ; answer moved to [2010h]
mov au, cu ; getting original value of au
mul bu; au*bu
mov [2020h], au; lower 16bit
mov [2022h], du; upper 16bit
mov du, 0000h ; resetting ds
mov au, cu
div bu ; au / bu
mov [2030h], au ; quotient
mov [2032h], du ; reminder
hlt
```

input: au = 3457h
bu = 0015h

output: [2000h] = sum = 346Ch
[2010h] = difference = 3442h
[2020h] = lower 16bit] = 044B23h
[2022h] = upper 16bit
[2030h] = quotient = 27Eh
[2032h] = reminder = 01h

Obj 2:

```
mov ax, 0000h
mov ds, ax
mov si, 2000h
mov al, [si]
rol al, 04h ; 4 times right shift
inc si ; si = 2001
mov [si], al ; al = data 1
mov di, 2010h
mov bl, [di] ; bl = data 2
mov cl, al ; cl = temp
and cl, bl ; cl = data1 and data 2
xor al, bl ; al = data1 xor data2
or al, cl ; al = al or cl
inc di
mov [di], al ; di = 2011
hlt ; result stored in 2011
```

input:
[2000h] = 21h
[2001h] = 34h

output:
[2002h] = 12h
[2012h] = 36h

Obj 3:

```
mov ax, 0000h
mov ds, ax
mov al, 12h
mov bl, al
shr al, 01h ; right shift by 1 time
xor al, bl
mov [2000h], al
hlt
```

input:
al = 12h

output:
[2000h] = 1Bh

Obj 4:

```
mov ax, 0000h
mov ds, ax
mov al, [2000h] ; input taken at [2000h]
not al ; compliment
inc al ; increment by 1
mov [2001h], al ; store answer
hlt
```

input:
[2000h] = 12h

output:
[2001h] = EEh

III. LAB

Note: For each objective do the following job and assessment:

- Screenshots of the Assembly language program (ALP)
- Observations (with screenshots)

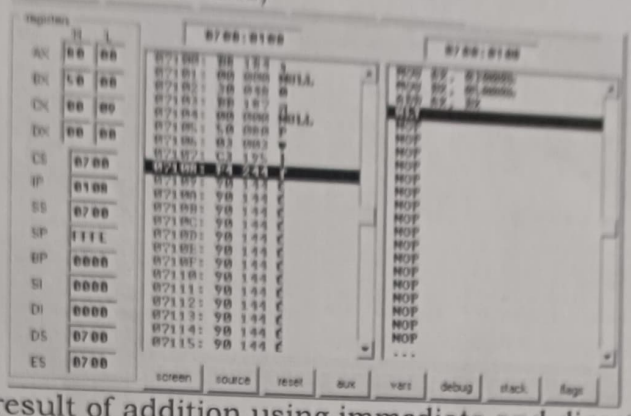
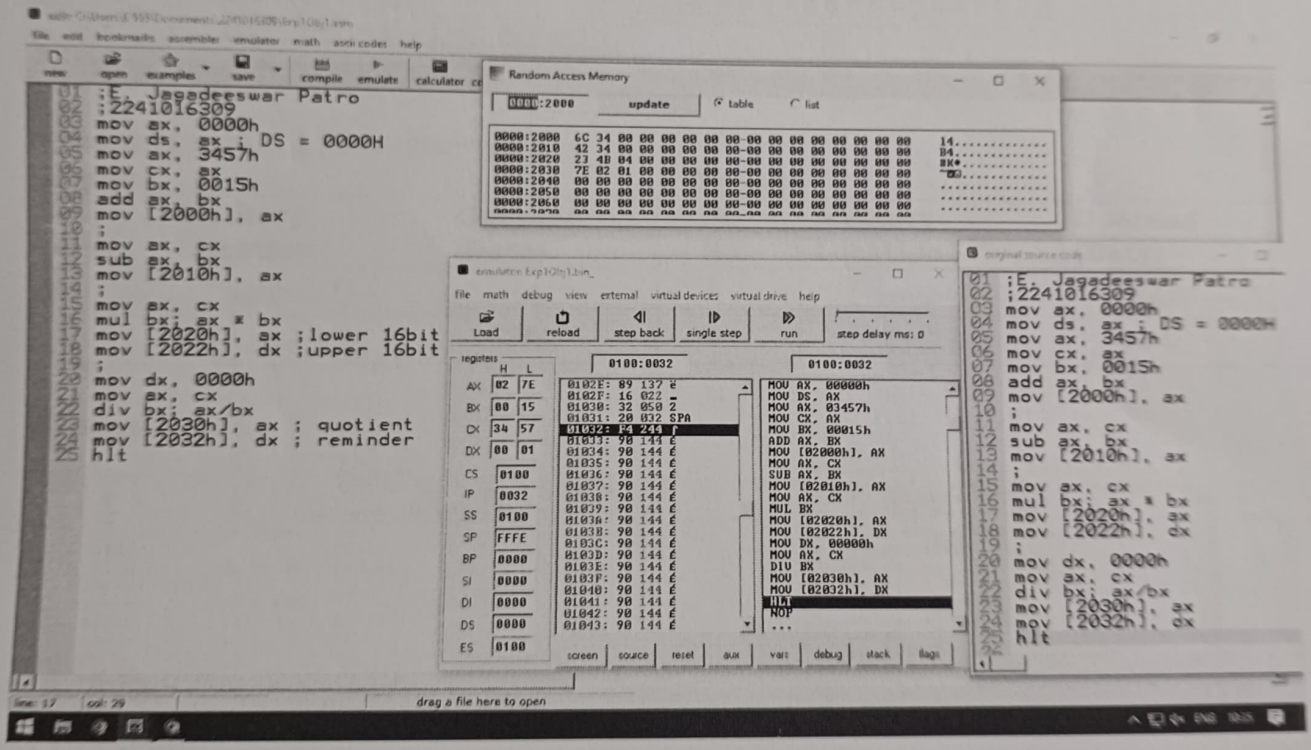


Fig. 1. Execution result of addition using immediate and direct addressing mode of 8086 emulator.

Objective 1:



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

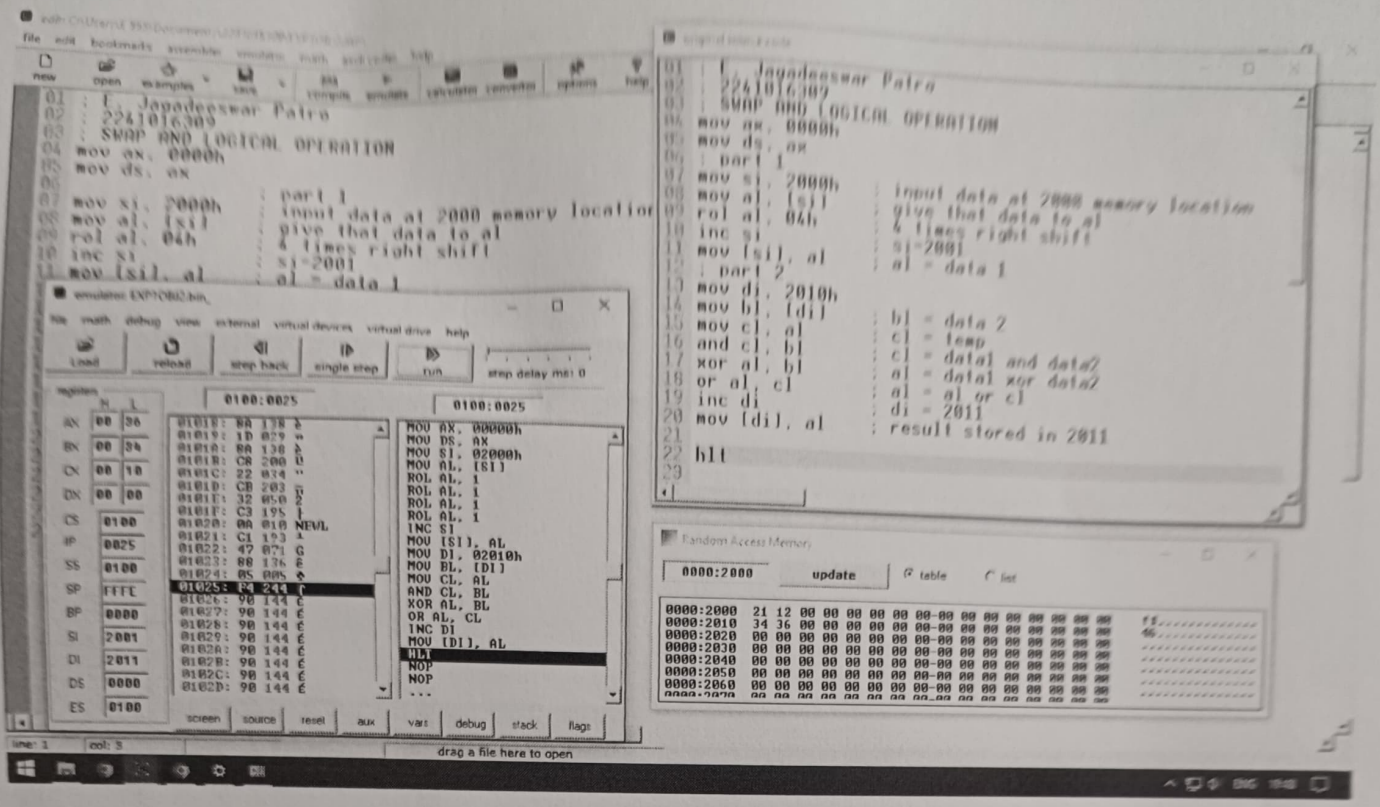
Input:

Sl. No.	Memory Location	Operand (Data)
1	ax	3457h
2	bx	0015h
3	cx	3457h

Output:

Sl. No.	Memory Location	Operand (Data)
1	[2000h]	346Ch
2	[2010h]	3442h
3	[2020h]	044B236h
4	[2030h]-[2032h]	27Eh, 01h

Objective 2:



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

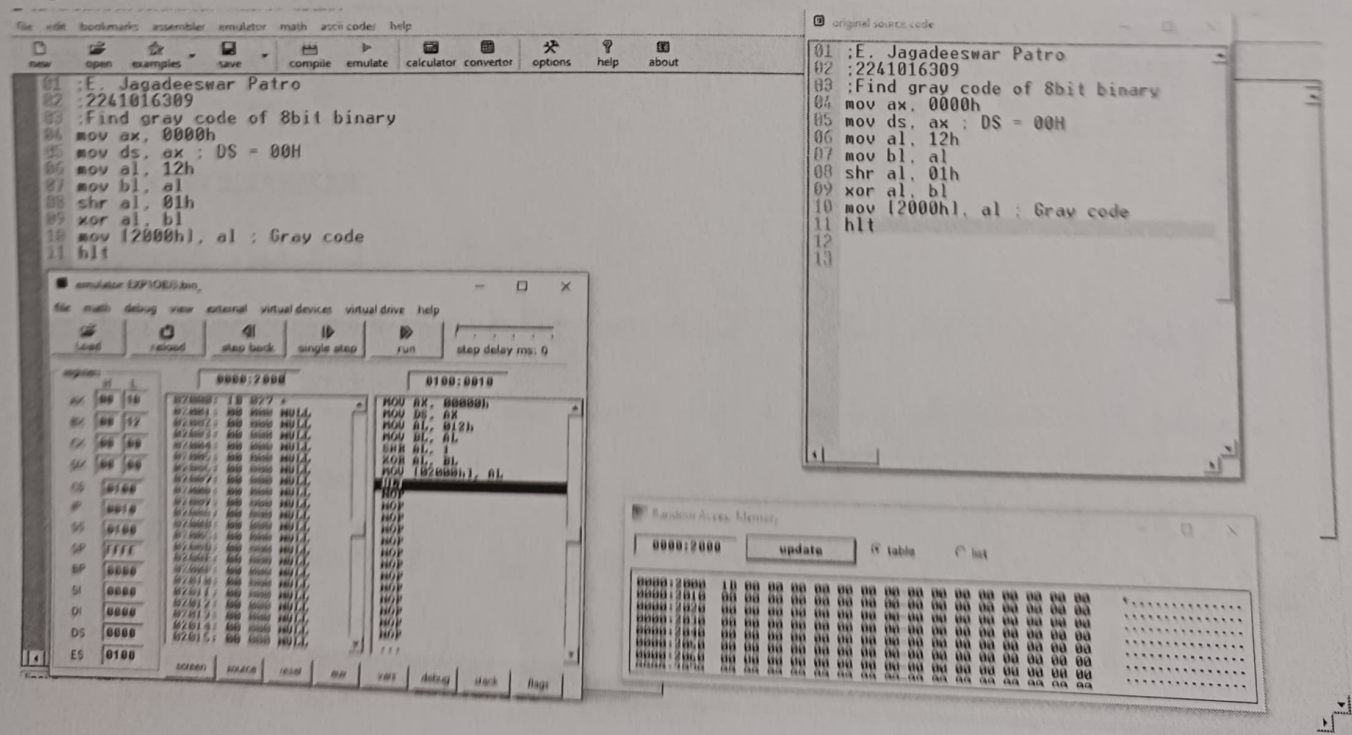
Input:

Sl. No.	Memory Location	Operand (Data)
1	[20004]	21h
2	[20014]	34h

Output:

Sl. No.	Memory Location	Operand (Data)
1	[20024]	12h
2	[20124]	36h

Objective 3:



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

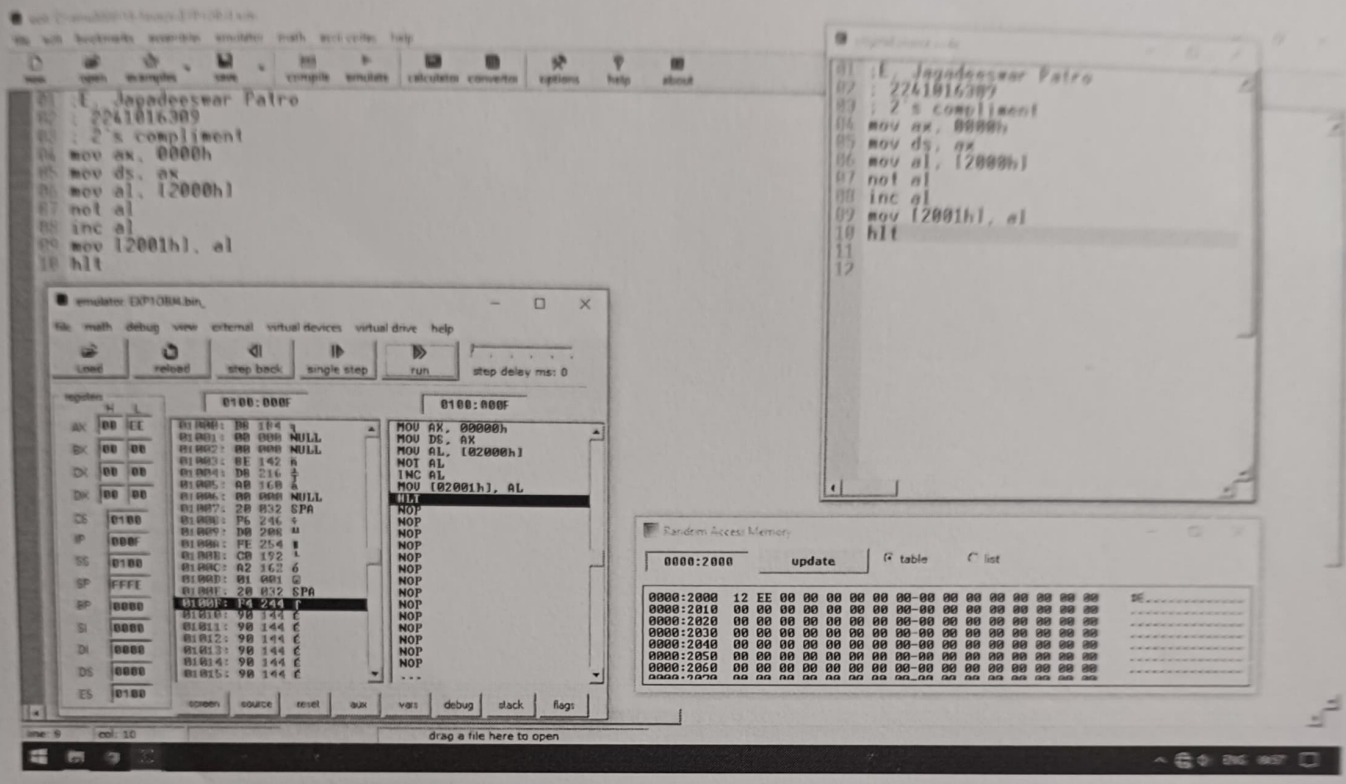
Input:

Sl. No.	Memory Location	Operand (Data)
1	al	12h

Output:

Sl. No.	Memory Location	Operand (Data)
1	[2000h]	1Bh

Objective 4:



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

Input:

Sl. No.	Memory Location	Operand (Data)
1	[2000h]	12h

Output:

Sl. No.	Memory Location	Operand (Data)
1	[2001h]	EEh

IV. CONCLUSION

It can be concluded that:

- immediate addressing takes operand in the instruction itself
- direct addressing takes operands as a 16-bit displacement element.
- indirect addressing takes operands placed in SI register
- add, sub, mul, div, shr, inc, dec, not, or, xor, and is used to perform the objectives and the output matches the predicted theoretical calculations.

V. POST LAB

1. Discuss different general-purpose registers used in 8086 microprocessors.
2. Explain the concept of segmented memory. What are its advantages?
3. Explain the physical address formation in 8086.
4. Write an assembly program to multiply 05H and 04H without using arithmetic instruction.
5. Write the function of the following logical instructions.
   a) SHL/SAL b) SHR c) SAR d) ROR e) ROL

Answers:

1. The Execution Unit (EU) has 8 general purpose registers and ~~two of them~~ can be paired to form 16-bit registers. The valid pairs are :-
   - AX (AL, AH): Word multiply, word divide, word I/O
   - BX (BL, BH): Store address information
   - CX (CL, CH): String operation, loops
   - DX (DL, DH): Word multiply, word divide, indirect I/O

2. Segmentation is the process in which the main memory of the computer is divided into different segments & each segment has its own base address.

   Advantages are:
   - provides powerfull memory mangement
   - Data related or stack related operations can be performed in different segments
   - Code related operation can be done in separate code segments.

3. Physical Address formation of 8086:

   Physical Address, PA = BA (Base Address) × 10h + offset

4.
```
mov ax, 0000h
mov ds, ax
mov ax, 05h ; input 2
shl ax, 02 ; left shift, each shift multiplies by 2.
mov [2000h], ax ; result stored at [2000h]
hlt
```

5. Logical instructions and their functions:

a) SHL/SAL :-
- Shift Logical / Arithmetic left
- Used to shift bits of a byte/word towards left and put zero(s) in LSBs.

b) SHR :-
- Shift Logical Right
- Used to shift bits of a byte/words towards right & put zero(s) in MSBs.

c) SAR :-
- Shift Arithmetic Right
- Used to shift bits of a byte/word towards right and copy the previous MSB into new MSB.

d) ROR :-
- Rotate Right without Carry
- Used to rotate bits of a byte/word towards the right i.e, LSB to MSB and to Carry Flag (CF).

e) ROL :-
- Rotate left without Carry
- Used to rotate bits of a byte/word towards left i.e. MSB to LSB and to Carry Flag (CF).