Computer Organization and Architecture (EET2211)

LAB I: Examine & Analyze Different Addressing Modes of 8086 Microprocessor

Siksha ‘O’ Anusandhan Deemed to be University, Bhubaneswar

Branch: Computer Science and Engineering		**Section:** ‘D’	
S. No.	**Name**	**Registration No.**	**Signature**
1	Saswat Mohanty	1941012407	Saswat Mohanty

Marks: ______/10

Remarks:

Teacher’s Signature

I. OBJECTIVE:

1. Addition of two 16bit numbers using immediate addressing mode.
2. Addition of two 16bit numbers using direct addressing mode.
3. Addition of two 16bit numbers using indirect addressing mode.
4. Addition of two 16bit numbers using index addressing mode.
5. Addition of two 16bit numbers using base index addressing mode.

II. PRE-LAB

For Obj. 1:

a) **Explain immediate addressing mode briefly.**

The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.

b) **Examine & analyze the output obtained from addition of two 16 bit numbers.**

MOV AX,2000H

MOV BX,9000H

ADD AX,BX

Output = B000h

c) **Write the assembly code.**

```
org 100h
MOV AX,2000H
MOV BX,9000H
ADD AX,BX
HLT
ret
```

For Obj. 2:

a) **Explain direct addressing mode briefly.**

The addressing mode in which the effective address of the memory location is written directly in the instruction

b) **Examine & analyze the output obtained from addition of two 16 bit numbers.**

mov ax,[2000h]

mov bx,[9000h]

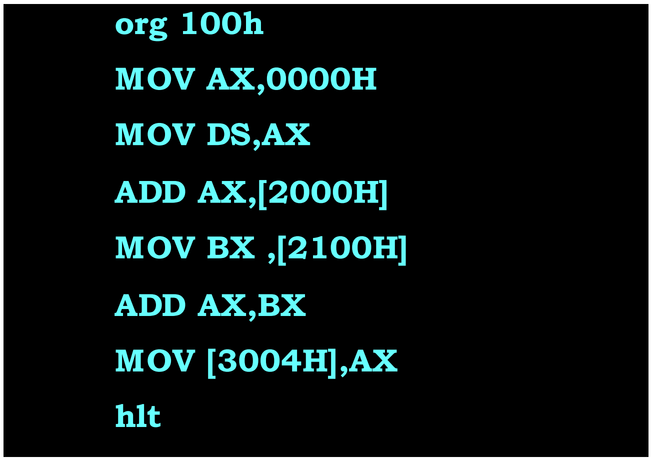
add ax,bx

[2000h] = 1111h

[9000h] = 2222h

Output: [3004h] = 3333h

c) **Write the assembly code.**



```
org 100h
MOV AX,0000H
MOV DS,AX
ADD AX,[2000H]
MOV BX ,[2100H]
ADD AX,BX
MOV [3004H],AX
hlt
```

For Obj. 3:

a) **Explain indirect addressing mode briefly.**

This addressing mode allows data to be addressed at any memory location through an offset address held in any of following registers BP, BX, DI and SI

b) Examine & analyze the output obtained from addition of two 16 bit numbers.

mov ax,[si]

mov bx,[si]

add ax,bx

[20400h] = 1111h

[20402h] = 2222h

Output : [20404] = 3333h

c) Write the assembly code.

```
org 100h
MOV AX,2000H
MOV DS,AX
MOV SI,0400H
MOV AX ,[SI]
INC SI
INC SI
MOV BX,[SI]
ADD AX,BX
INC SI
INC SI
MOV [SI],AX
hlt
```

For Obj. 4:

a) Explain index addressing mode briefly.

In this addressing mode, the operands offset address is found by adding the contents of SI or DI register and 8 bit/ 16 bit displacements

b) **Examine & analyze the output obtained from addition of two 16 bit numbers.**

mov ax,[si]

mov bx,[si+2]

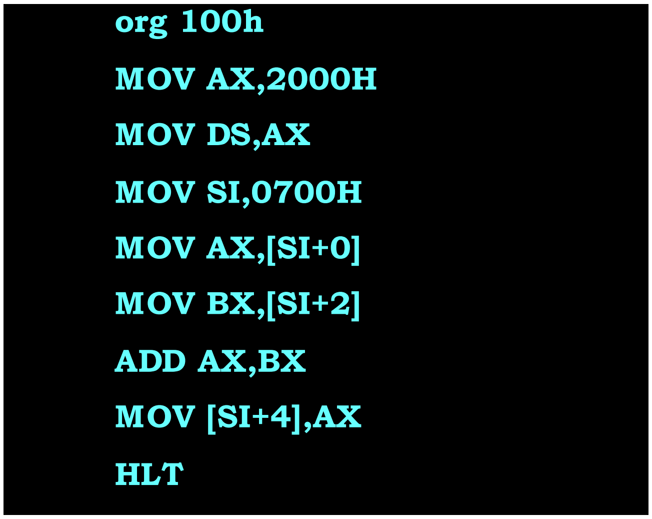
add ax,bx

[20700h] = 1111h

[20702h] = 2222h

Output: [20704] = 3333h

c) **Write the assembly code.**



```
org 100h
MOV AX,2000H
MOV DS,AX
MOV SI,0700H
MOV AX,[SI+0]
MOV BX,[SI+2]
ADD AX,BX
MOV [SI+4],AX
HLT
```

For Obj. 5:

a) **Explain base index addressing mode briefly.**

In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register.

b) **Examine & analyze the output obtained from addition of two 16 bit numbers.**

mov ax,[bx+si]

mov cx,[bx+si]

add ax,cx

[0000h] = 1111h

[3500h] = 2222h

[3502h] = 3333h

Output: [3504] = 5555h

c) Write the assembly code.

```
org 100h
MOV AX,0000H
MOV DS,AX
MOV BX,3000H
MOV SI,0500H
MOV CX,[BX+SI]
MOV DX,[BX+SI+02]
MOV AX,CX
ADD AX,DX
HLT
```

III. LAB:

Assembly Program:

For Obj. 1

```
; SASWAT MOHANTY
; 1941012407

; Addition of two 16bit numbers using immediate addressing mode

org 100h

MOV AX,2000H
MOV BX,9000H
ADD AX,BX
HLT
```

```
ret
```

For Obj. 2

```
; SASWAT MOHANTY
; 1941012407

; Addition  of two 16bit numbers using direct addressing mode

org 100h

MOV AX,0000H
MOV DS,AX
ADD AX,[2000H]      ; value stored at 2000 = 1111
MOV BX ,[2100H]     ; value stored at 2100 = 2222
ADD AX,BX
MOV [3004H],AX

hlt

ret
```

For Obj. 3

```
; SASWAT MOHANTY
; 1941012407

; Addition  of two 16bit numbers using indirect addressing mode

org 100h

MOV AX,2000H
MOV DS,AX
MOV SI,0400H
MOV AX ,[SI]         ; value stored at 20400 = 1111
INC SI               ; value stored at 20402 = 2222
INC SI
MOV BX,[SI]
ADD AX,BX
INC SI
INC SI
MOV [SI],AX
```

```
hlt

ret
```

For Obj. 4

```
; SASWAT MOHANTY
; 1941012407

; Addition  of two 16bit numbers using index addressing mode

org 100h

MOV AX,2000H
MOV DS,AX
MOV SI,0700H          ;VALUES STORED AT 20700 = 1111
MOV AX,[SI+0]         ;VALUES STORED AT 20702 = 2222
MOV BX,[SI+2]
ADD AX,BX
MOV [SI+4],AX

HLT

ret
```

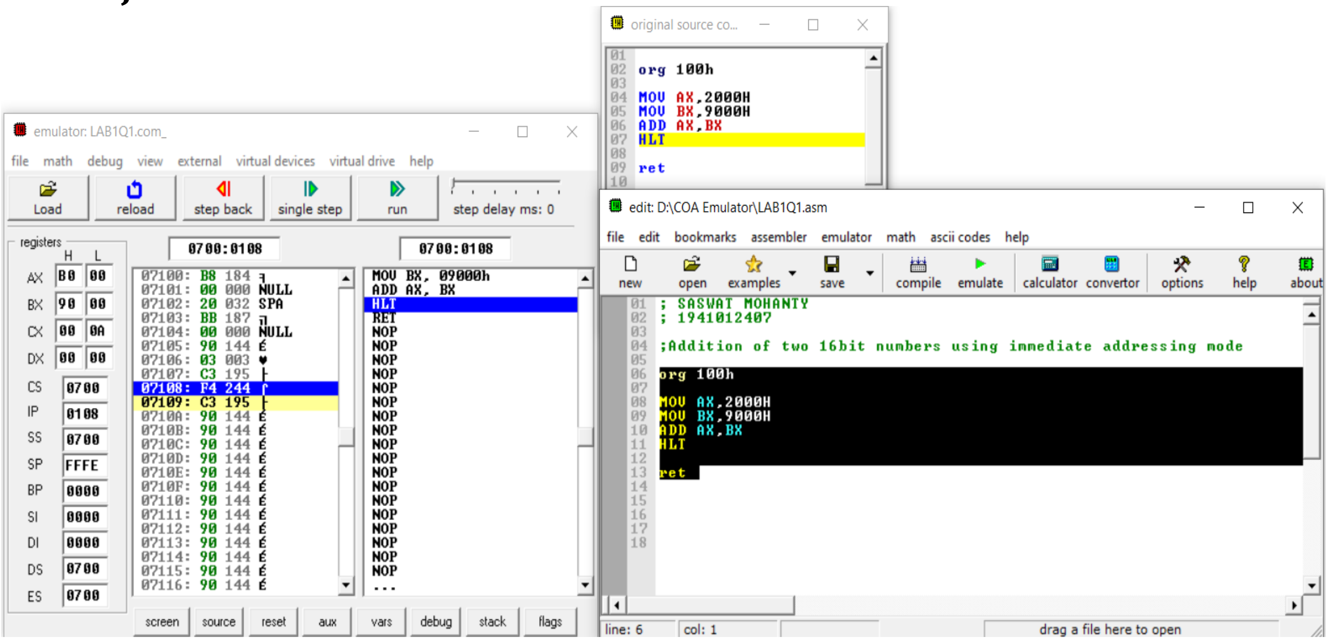
For Obj. 5

```
; SASWAT MOHANTY
; 1941012407

; Addition  of two 16bit numbers using base index addressing mode

org 100h

MOV AX,0000H               ;value stored at 0000 = 1111
MOV DS,AX
MOV BX,3000H
MOV SI,0500H               ;value stored at 3500 = 2222
MOV CX,[BX+SI]             ;value stored at 3502 = 3333
MOV DX,[BX+SI+02]
MOV AX,CX
ADD AX,DX

HLT
```

```
ret
```

Observations (with screen shots):

For Obj. 1


original source co...
emulator: LAB1Q1.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
edit: D:\COA Emulator\LAB1Q1.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
drag a file here to open

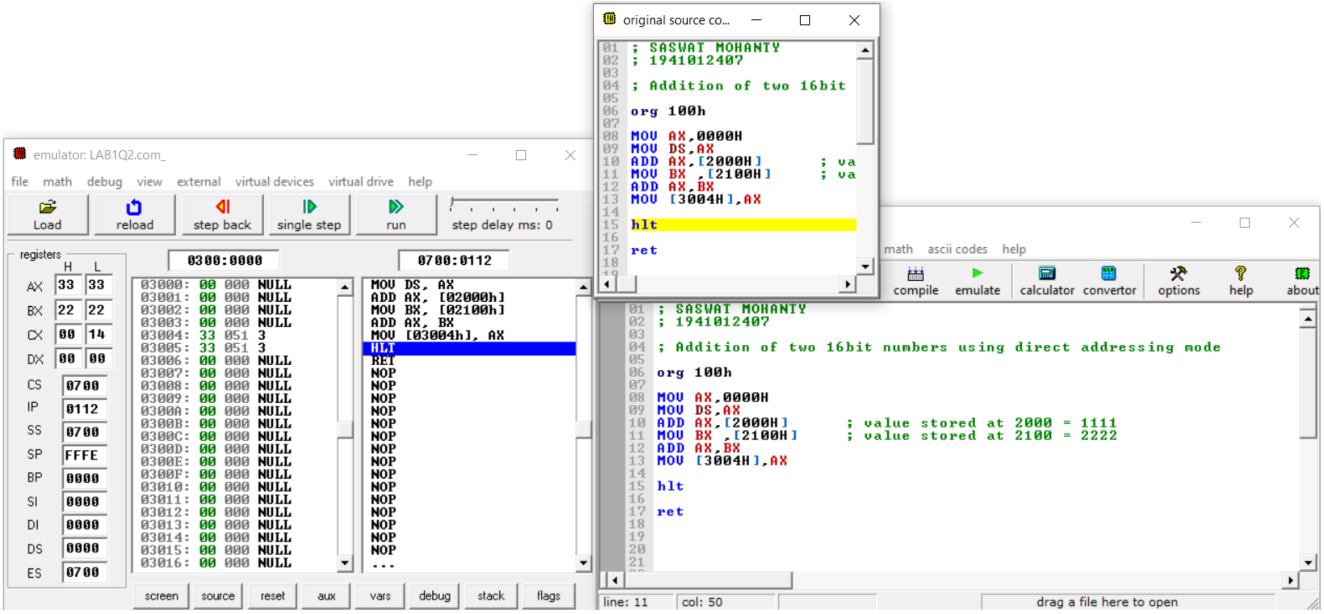
```
org 100h

MOV AX,2000H
MOV BX,9000H
ADD AX,BX
HLT

ret
```

```
registers
     H   L
AX   B0  00
BX   90  00
CX   00  0A
DX   00  00
CS   0700
IP   0108
SS   0700
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700

0700:0108                    0700:0108
07100: B8 184 ╕              MOV BX, 09000h
07101: 00 000 NULL           ADD AX, BX
07102: 20 032 SPA            HLT
07103: BB 187 ╗              RET
07104: 00 000 NULL           NOP
07105: 90 144 É              NOP
07106: 03 003 ♥              NOP
07107: C3 195 ├              NOP
07108: F4 244 ⌠              NOP
07109: C3 195 ├              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              ...
07116: 90 144 É
```

```
; SASWAT MOHANTY
; 1941012407

;Addition of two 16bit numbers using immediate addressing mode

org 100h

MOV AX,2000H
MOV BX,9000H
ADD AX,BX
HLT

ret
```

line: 6 col: 1

For Obj. 2


original source co...
emulator: LAB1Q2.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
math ascii codes help
compile emulate calculator convertor options help about
drag a file here to open

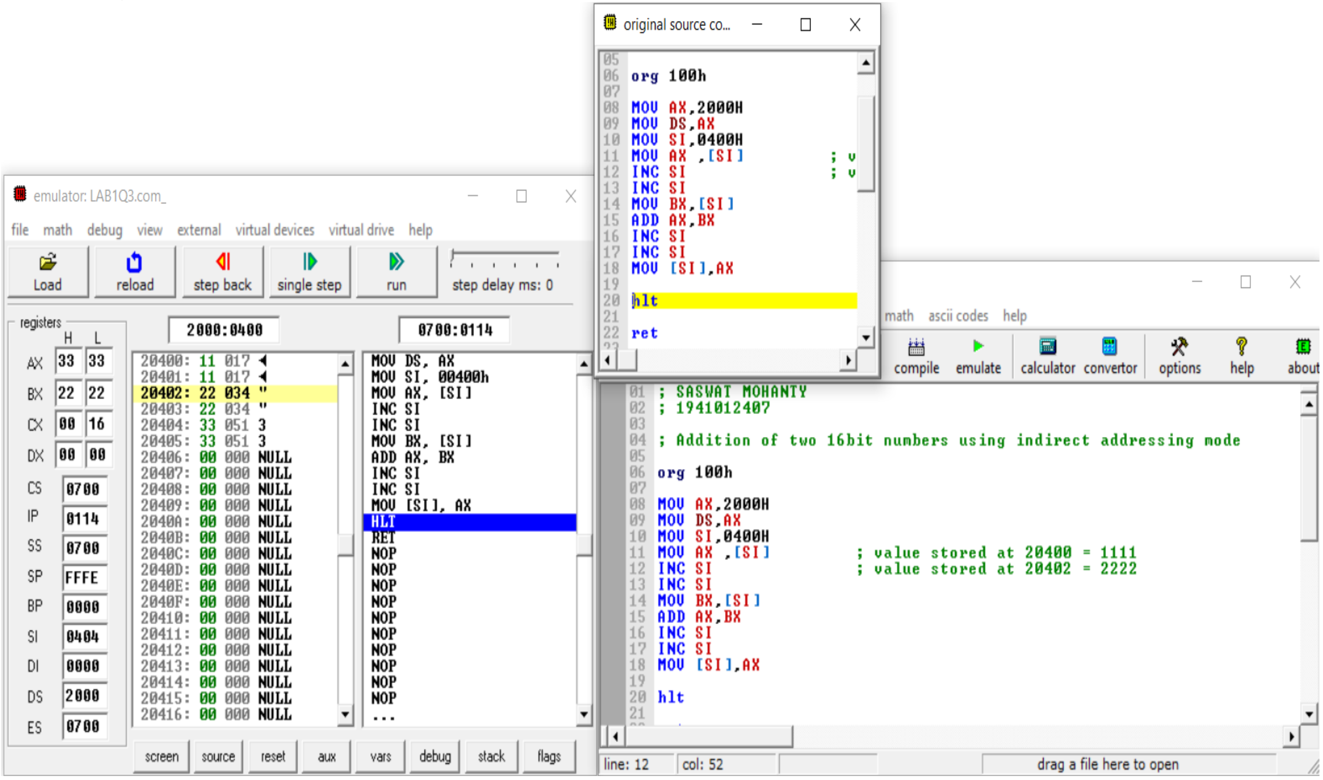
```
; SASWAT MOHANTY
; 1941012407

; Addition of two 16bit

org 100h

MOV AX,0000H
MOV DS,AX
ADD AX,[2000H]        ; va
MOV BX ,[2100H]       ; va
ADD AX,BX
MOV [3004H],AX

hlt

ret
```

```
registers
     H   L
AX   33  33
BX   22  22
CX   00  14
DX   00  00
CS   0700
IP   0112
SS   0700
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0000
ES   0700

0300:0000                    0700:0112
03000: 00 000 NULL           MOV DS, AX
03001: 00 000 NULL           ADD AX, [02000h]
03002: 00 000 NULL           MOV BX, [02100h]
03003: 00 000 NULL           ADD AX, BX
03004: 33 051 3              MOV [03004h], AX
03005: 33 051 3              HLT
03006: 00 000 NULL           RET
03007: 00 000 NULL           NOP
03008: 00 000 NULL           NOP
03009: 00 000 NULL           NOP
0300A: 00 000 NULL           NOP
0300B: 00 000 NULL           NOP
0300C: 00 000 NULL           NOP
0300D: 00 000 NULL           NOP
0300E: 00 000 NULL           NOP
0300F: 00 000 NULL           NOP
03010: 00 000 NULL           NOP
03011: 00 000 NULL           NOP
03012: 00 000 NULL           NOP
03013: 00 000 NULL           NOP
03014: 00 000 NULL           NOP
03015: 00 000 NULL           NOP
03016: 00 000 NULL           ...
```

```
; SASWAT MOHANTY
; 1941012407

; Addition of two 16bit numbers using direct addressing mode

org 100h

MOV AX,0000H
MOV DS,AX
ADD AX,[2000H]        ; value stored at 2000 = 1111
MOV BX ,[2100H]       ; value stored at 2100 = 2222
ADD AX,BX
MOV [3004H],AX

hlt

ret
```

line: 11 col: 50

For Obj. 3



original source co...
emulator: LAB1Q3.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
screen source reset aux vars debug stack flags
math ascii codes help
compile emulate calculator convertor options help about
drag a file here to open

```
org 100h

MOV AX,2000H
MOV DS,AX
MOV SI,0400H
MOV AX ,[SI]           ; v
INC SI                 ; v
INC SI
MOV BX,[SI]
ADD AX,BX
INC SI
INC SI
MOV [SI],AX

hlt

ret
```

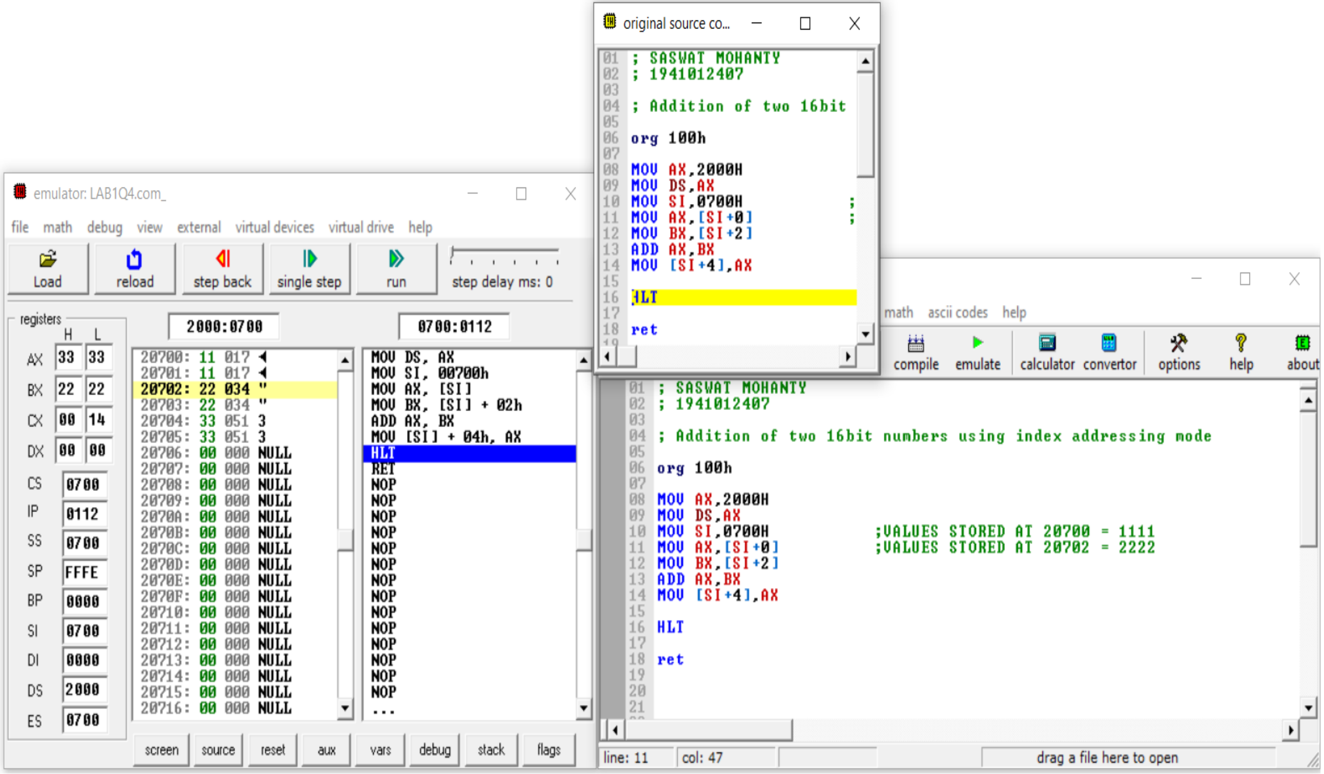
	H	L
AX	33	33
BX	22	22
CX	00	16
DX	00	00
CS	0700	
IP	0114	
SS	0700	
SP	FFFE	
BP	0000	
SI	0404	
DI	0000	
DS	2000	
ES	0700	

```
2000:0400                   0700:0114
20400: 11 017 ◄             MOV DS, AX
20401: 11 017 ◄             MOV SI, 00400h
20402: 22 034 "             MOV AX, [SI]
20403: 22 034 "             INC SI
20404: 33 051 3             INC SI
20405: 33 051 3             MOV BX, [SI]
20406: 00 000 NULL          ADD AX, BX
20407: 00 000 NULL          INC SI
20408: 00 000 NULL          INC SI
20409: 00 000 NULL          MOV [SI], AX
2040A: 00 000 NULL          HLT
2040B: 00 000 NULL          RET
2040C: 00 000 NULL          NOP
2040D: 00 000 NULL          NOP
2040E: 00 000 NULL          NOP
2040F: 00 000 NULL          NOP
20410: 00 000 NULL          NOP
20411: 00 000 NULL          NOP
20412: 00 000 NULL          NOP
20413: 00 000 NULL          NOP
20414: 00 000 NULL          NOP
20415: 00 000 NULL          NOP
20416: 00 000 NULL          ...
```

```
; SASWAT MOHANTY
; 1941012407

; Addition of two 16bit numbers using indirect addressing mode

org 100h

MOV AX,2000H
MOV DS,AX
MOV SI,0400H
MOV AX ,[SI]           ; value stored at 20400 = 1111
INC SI                 ; value stored at 20402 = 2222
INC SI
MOV BX,[SI]
ADD AX,BX
INC SI
INC SI
MOV [SI],AX

hlt

```

line: 12 col: 52

For Obj. 4



original source co...
emulator: LAB1Q4.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
screen source reset aux vars debug stack flags
math ascii codes help
compile emulate calculator convertor options help about
drag a file here to open

```
; SASWAT MOHANTY
; 1941012407

; Addition of two 16bit

org 100h

MOV AX,2000H
MOV DS,AX
MOV SI,0700H           ;
MOV AX,[SI+0]          ;
MOV BX,[SI+2]
ADD AX,BX
MOV [SI+4],AX

HLT

ret
```

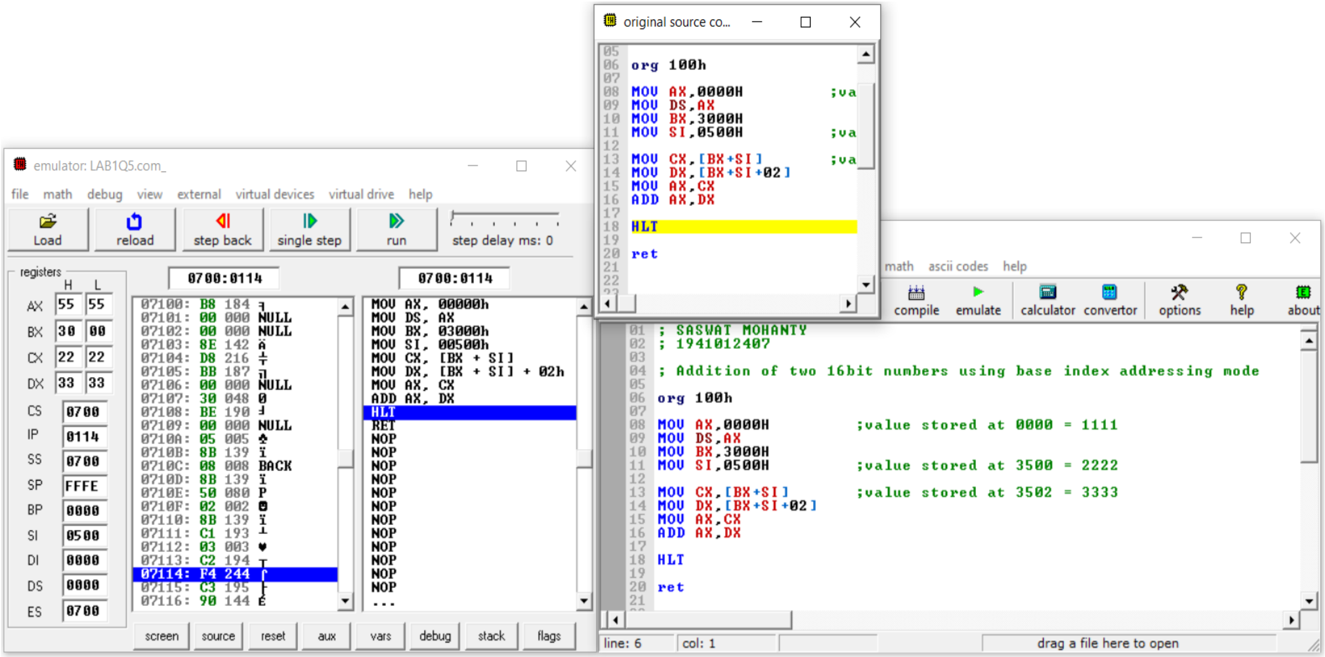
	H	L
AX	33	33
BX	22	22
CX	00	14
DX	00	00
CS	0700	
IP	0112	
SS	0700	
SP	FFFE	
BP	0000	
SI	0700	
DI	0000	
DS	2000	
ES	0700	

```
2000:0700                   0700:0112
20700: 11 017 ◄             MOV DS, AX
20701: 11 017 ◄             MOV SI, 00700h
20702: 22 034 "             MOV AX, [SI]
20703: 22 034 "             MOV BX, [SI] + 02h
20704: 33 051 3             ADD AX, BX
20705: 33 051 3             MOV [SI] + 04h, AX
20706: 00 000 NULL          HLT
20707: 00 000 NULL          RET
20708: 00 000 NULL          NOP
20709: 00 000 NULL          NOP
2070A: 00 000 NULL          NOP
2070B: 00 000 NULL          NOP
2070C: 00 000 NULL          NOP
2070D: 00 000 NULL          NOP
2070E: 00 000 NULL          NOP
2070F: 00 000 NULL          NOP
20710: 00 000 NULL          NOP
20711: 00 000 NULL          NOP
20712: 00 000 NULL          NOP
20713: 00 000 NULL          NOP
20714: 00 000 NULL          NOP
20715: 00 000 NULL          NOP
20716: 00 000 NULL          ...
```

```
; SASWAT MOHANTY
; 1941012407

; Addition of two 16bit numbers using index addressing mode

org 100h

MOV AX,2000H
MOV DS,AX
MOV SI,0700H                   ;VALUES STORED AT 20700 = 1111
MOV AX,[SI+0]                  ;VALUES STORED AT 20702 = 2222
MOV BX,[SI+2]
ADD AX,BX
MOV [SI+4],AX

HLT

ret

```

line: 11 col: 47

For Obj. 5



registers	H	L
AX	55	55
BX	30	00
CX	22	22
DX	33	33
CS	0700	
IP	0114	
SS	0700	
SP	FFFE	
BP	0000	
SI	0500	
DI	0000	
DS	0000	
ES	0700	

```
0700:0114
07100: B8 184 ╕
07101: 00 000 NULL
07102: 00 000 NULL
07103: 8E 142 Ä
07104: D8 216 ╪
07105: BB 187 ╗
07106: 00 000 NULL
07107: 30 048 0
07108: BE 190 ╛
07109: 00 000 NULL
0710A: 05 005 ♣
0710B: 8B 139 ï
0710C: 08 008 BACK
0710D: 8B 139 ï
0710E: 50 080 P
0710F: 02 002 ☻
07110: 8B 139 ï
07111: C1 193 ┴
07112: 03 003 ♥
07113: C2 194 ┬
07114: F4 244 ⌠
07115: C3 195 ├
07116: 90 144 É
```

```
0700:0114
MOV AX, 00000h
MOV DS, AX
MOV BX, 03000h
MOV SI, 00500h
MOV CX, [BX + SI]
MOV DX, [BX + SI] + 02h
MOV AX, CX
ADD AX, DX
HLT
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

```

org 100h

MOV AX,0000H            ;va
MOV DS,AX
MOV BX,3000H
MOV SI,0500H            ;va

MOV CX,[BX+SI]          ;va
MOV DX,[BX+SI+02]
MOV AX,CX
ADD AX,DX

HLT

ret
```

```
; SASWAT MOHANTY
; 1941012407

; Addition of two 16bit numbers using base index addressing mode

org 100h

MOV AX,0000H            ;value stored at 0000 = 1111
MOV DS,AX
MOV BX,3000H
MOV SI,0500H            ;value stored at 3500 = 2222

MOV CX,[BX+SI]          ;value stored at 3502 = 3333
MOV DX,[BX+SI+02]
MOV AX,CX
ADD AX,DX

HLT

ret
```

Conclusion:

For Obj. 1:

It can be concluded that for immediate addressing the operand is specified in the instruction itself.

For Obj. 2:

It can be concluded that for direct addressing the operands offset is given in the instruction as a 16-bit displacement element.

For Obj. 3:

It can be concluded that for indirect addressing the operands offset is placed SI register as specified in the instruction.

For Obj. 4:

It can be concluded that for index addressing the offset is the sum of the content of SI register and a 16-bit displacement element.

For Obj. 5:

It can be concluded that for base index addressing the offset is the sum of the content of BX and SI register.

IV. POST LAB:

1. Discuss different general-purpose registers used in 8086 microprocessor.

EU has 8 general purpose registers; two registers can also be combined to form 16-bit registers. The valid register 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 (used to hold I/O address during I/O instructions If the result is more than 16 bits, the lower order 16 bits are stored in accumulator and higher order 6 bits are stored in DX register)

2. Explain the concept of segmented memory. What are its advantages?

Segmentation is the process in which the main memory of the computer is divided into different segments and each segment has its own base address. It is basically used to enhance the speed of execution of the computer system, so that processor is able to fetch and execute the data from the memory easily and fast.

The main advantages of segmentation memory are as follows:

1) It provides a powerful memory management mechanism.
2) Data related or stack related operations can be performed in different segments.
3) Code related operation can be done in separate code segments.
4) It allows to processes to easily share data.
5) It allows extending the address ability of the processor, i.e., segmentation allows the use of 16-bit registers to give an addressing capability of 1 Megabytes. Without segmentation, it would require 20-bit registers.

6) It is possible to enhance the memory size of code data or stack segments beyond 64 KB by allotting more than one segment for each area.

3. Explain the physical address formation in 8086.

Physical Address = Base Address * 10H + Offset

4. Write a program to add two 16 bit numbers 12H and 08H, and store the sum.

```
org 100h
mov ax,0012h
mov bx,0008h
add ax,bx
hlt
```