SIKSHA 'O' ANUSANDHAN

INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH

Computer Organisation and Architecture (EET2211)

LAB REPORTS

SUBMITTED BY:-

NAME: - Saswat Mohanty

REGD NO.: - 1941012407

BRANCH: - CSE

SECTION: - 'D'

SEMESTER: - 4th

PROGRAMME: - B. TECH

CONTENTS

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**
52	Saswat Mohanty	1941012407	

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
```

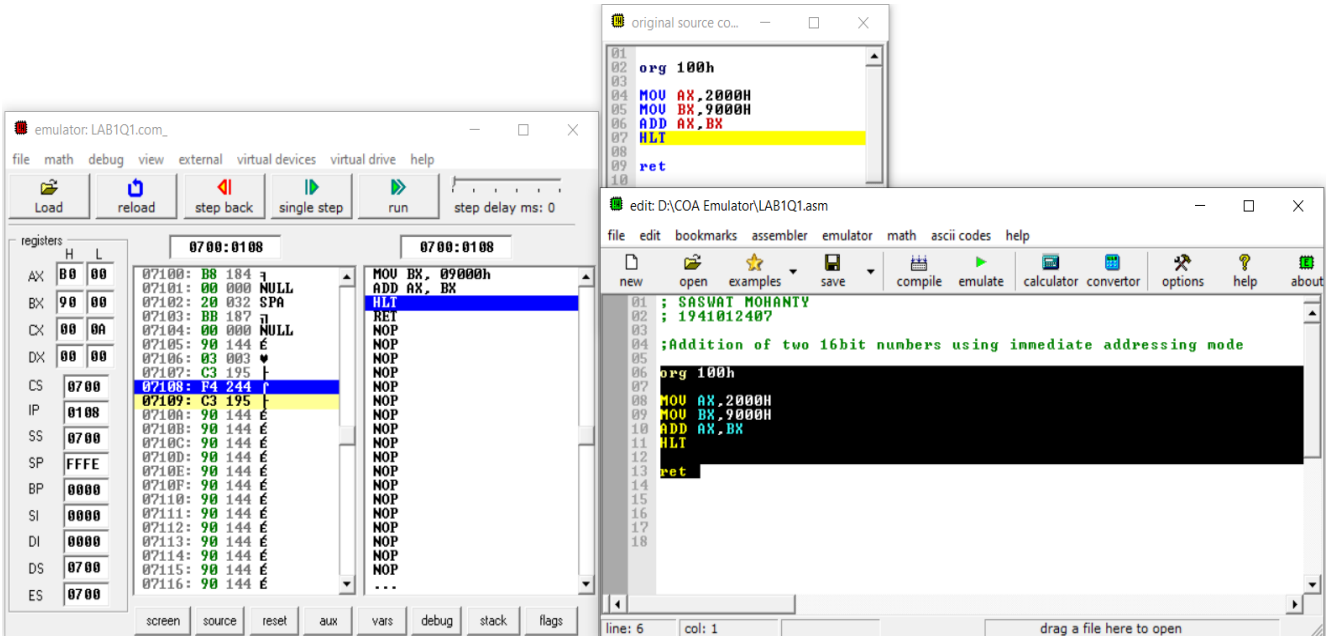
For Obj. 5

```
; SASWAT MOHANTY
; 1941012407

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

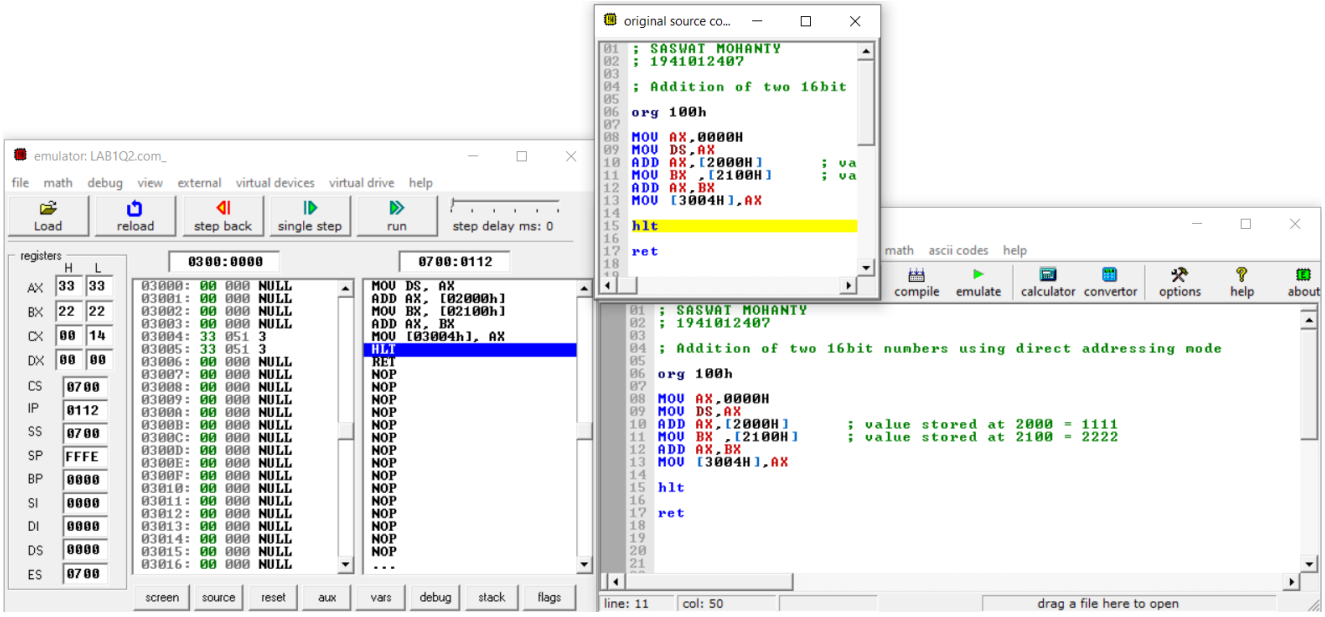
```
org 100h

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):

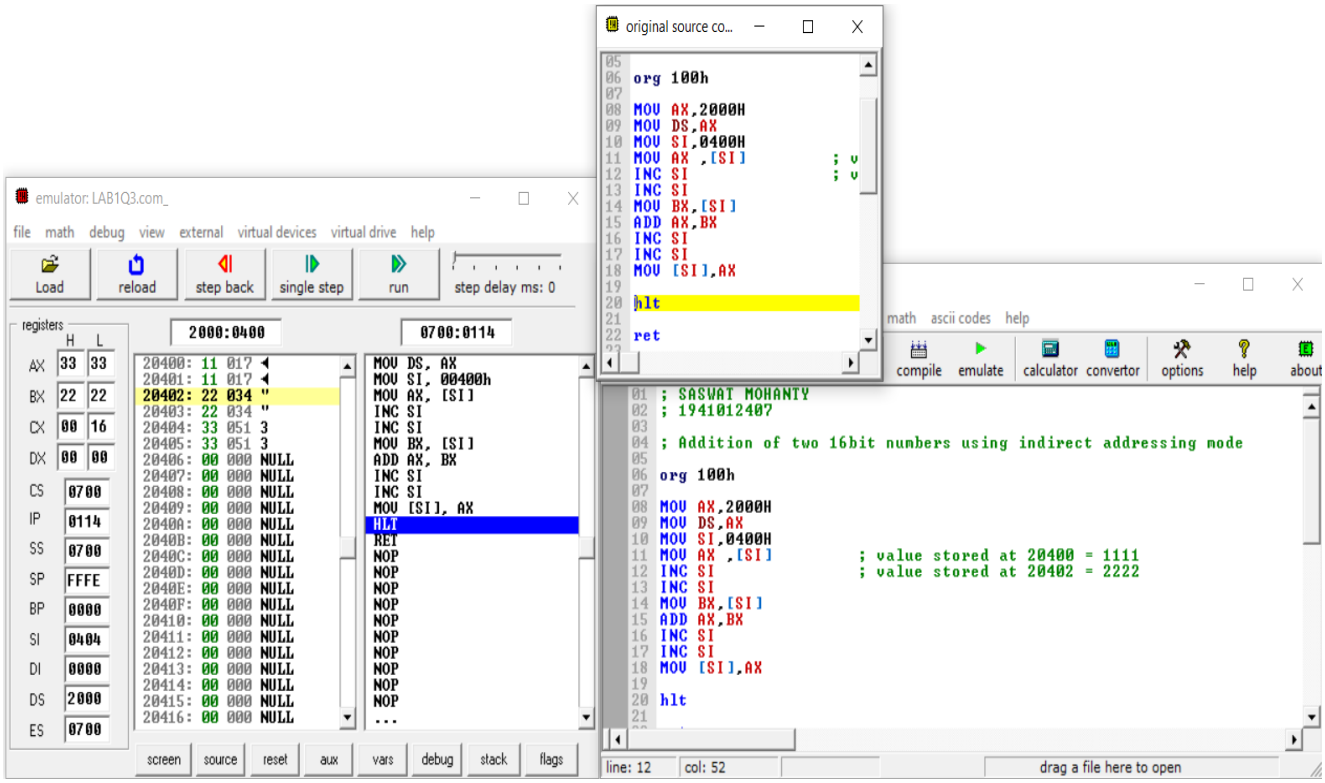
For Obj. 1



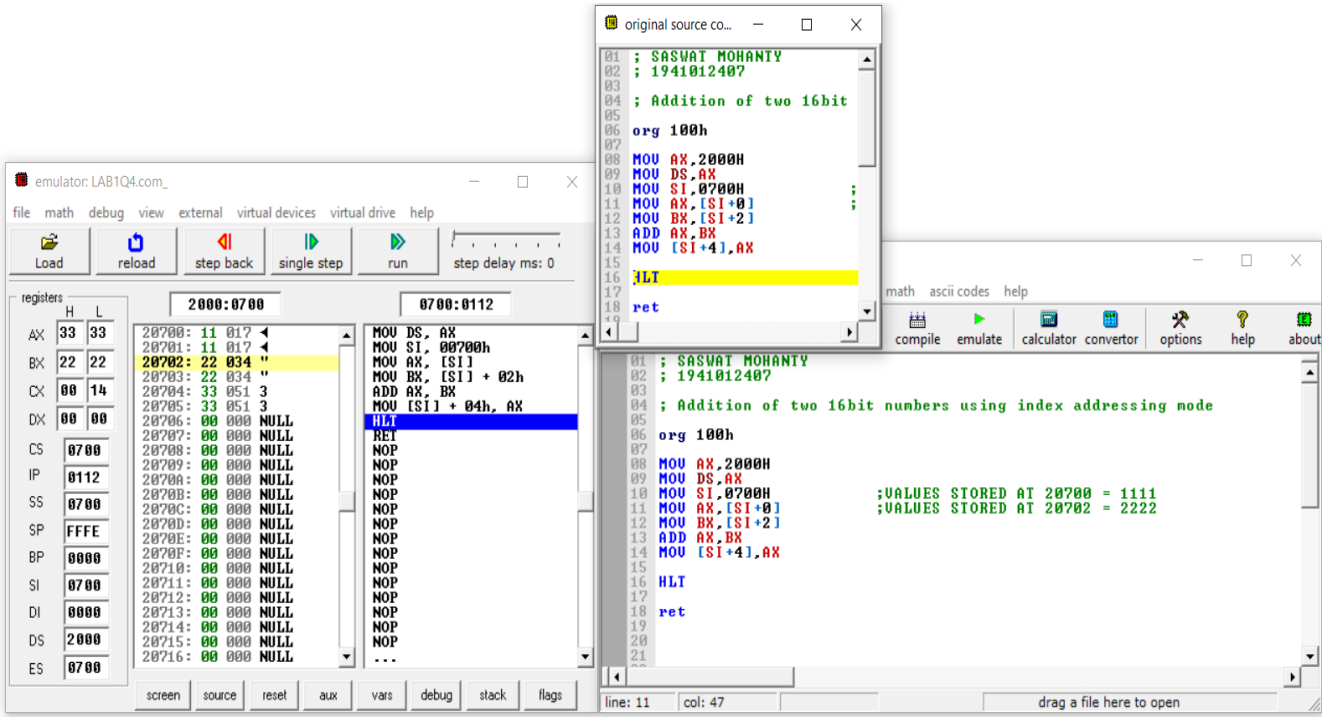
For Obj. 2


original source co...
; 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
emulator: LAB1Q2.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
0300:0000
0700:0112
compile
emulate
calculator
convertor
options
help
about
; 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
screen
source
reset
aux
vars
debug
stack
flags
line: 11
col: 50
drag a file here to open

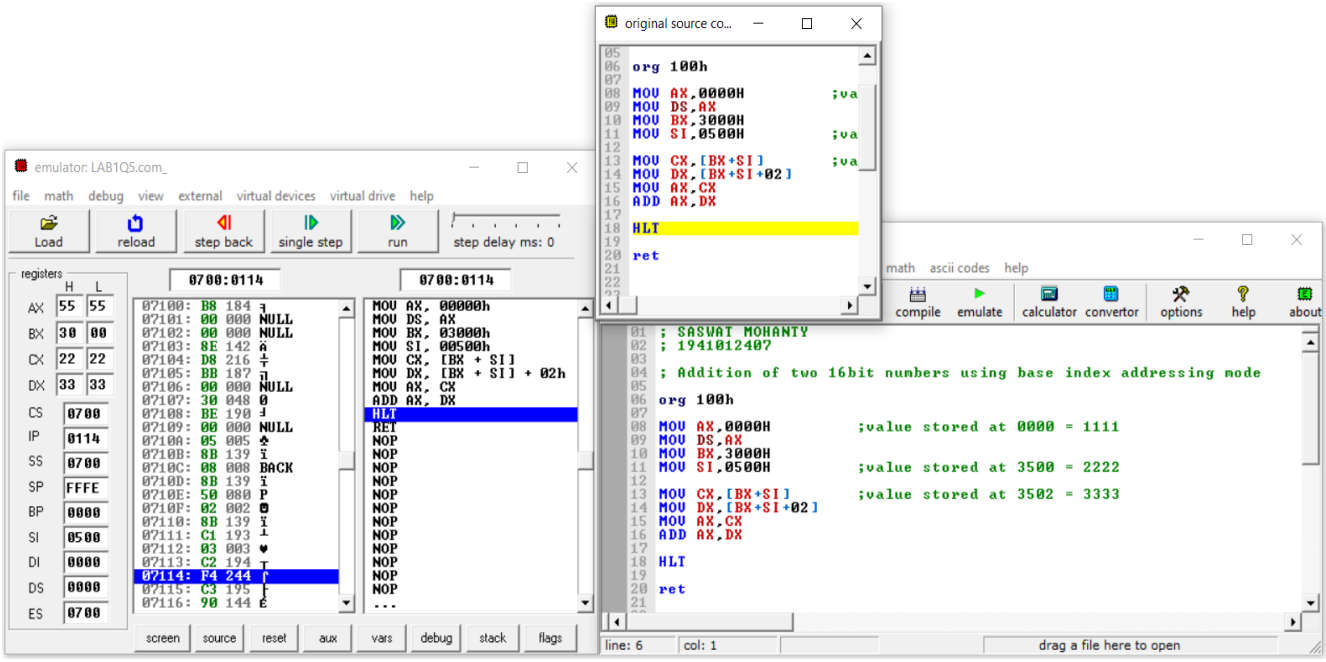
For Obj. 3


original source co...
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
ret
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
2000:0400
0700:0114
compile
emulate
calculator
convertor
options
help
about
; 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
screen
source
reset
aux
vars
debug
stack
flags
line: 12
col: 52
drag a file here to open

For Obj. 4



For Obj. 5



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
```

Computer Organization and Architecture (EET2211)

LAB II: Evaluate Different Arithmetic Operations on two 16 bit data

Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar

Branch: Computer Science and Engineering			**Section:** 'D'
S. No.	**Name**	**Registration No.**	**Signature**
52	Saswat Mohanty	1941012407	Saswat Mohanty

Marks: _____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1. Addition of two 16 bit numbers using direct addressing mode.
2. Subtraction of two 16 bit numbers using direct addressing mode.
3. Multiplication of two 16 bit numbers using direct addressing mode.
4. Division of two 16 bit numbers using direct addressing mode.

II. PRE-LAB

For Obj. 1:

a) Explain direct addressing mode briefly.

It is 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,[1000h]

mov bx,[1002h]

add ax,bx

[1000h] = 1111h

[1002h] = 1234h

Output: 2345h

c) Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov ax,[5000h]
```

```
mov bx,[5002h]
add ax,bx
mov [5004h],ax
hlt
```

For Obj. 2:

a) Examine & analyze the output obtained from subtraction of two 16 bit numbers.

mov ax,[1000h]

mov bx,[1002h]

sub ax,bx

[1000h] = 2222h

[1002h] = 1111h

Output: 1111h

b) Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov ax,[3000h]
mov bx,[3002h]
sub ax,bx
mov [3004h],ax
hlt
```

For Obj. 3:

a) Examine & analyze the output obtained from multiplication of two 16 bit numbers.

mov ax,[1000h]

mov bx,[1002h]

mul bx,ax

[1000h] = 2222h

[1002h] = 1111h

Output: 2468642h

b) Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov ax,[3000h]
mov bx,[3002h]
mul bx
mov [3004h],ax
mov [3006h],dx
hlt
```

For Obj. 4:

a) Examine & analyze the output obtained from division of two 16 bit numbers.

mov ax,[1000h]

mov bx,[1002h]

div bx

[1000h] = 2222h

[1002h] = 1111h

Output: 2h

b) Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov ax,[3000h]
mov bx,[3002h]
div bx
mov [3004h],ax
mov [3006h],dx
hlt
ret
```

III. LAB:

Assembly Program:

For Obj. 1

```
; Saswat Mohanty
; 1941012407

; Addition of two 16 bit numbers using direct addressing mode

org 100h

mov ax,0000h
mov ds,ax
mov ax,[5000h]      ;VALUE STORED AT 5000 = 1111
mov bx,[5002h]      ;VALUE STORED AT 5002 = 2222
add ax,bx
```

```
mov [5004h],ax
hlt

ret
```

For Obj. 2

```
; Saswat Mohanty
; 1941012407

; Subtraction of two 16 bit numbers using direct addressing mode

org 100h

mov ax,0000h
mov ds,ax
mov ax,[3000h]      ;VALUE STORED AT 3000 = 2222
mov bx,[3002h]      ;VALUE STORED AT 3002 = 1111
sub ax,bx
mov [3004h],ax
hlt

ret
```

For Obj. 3

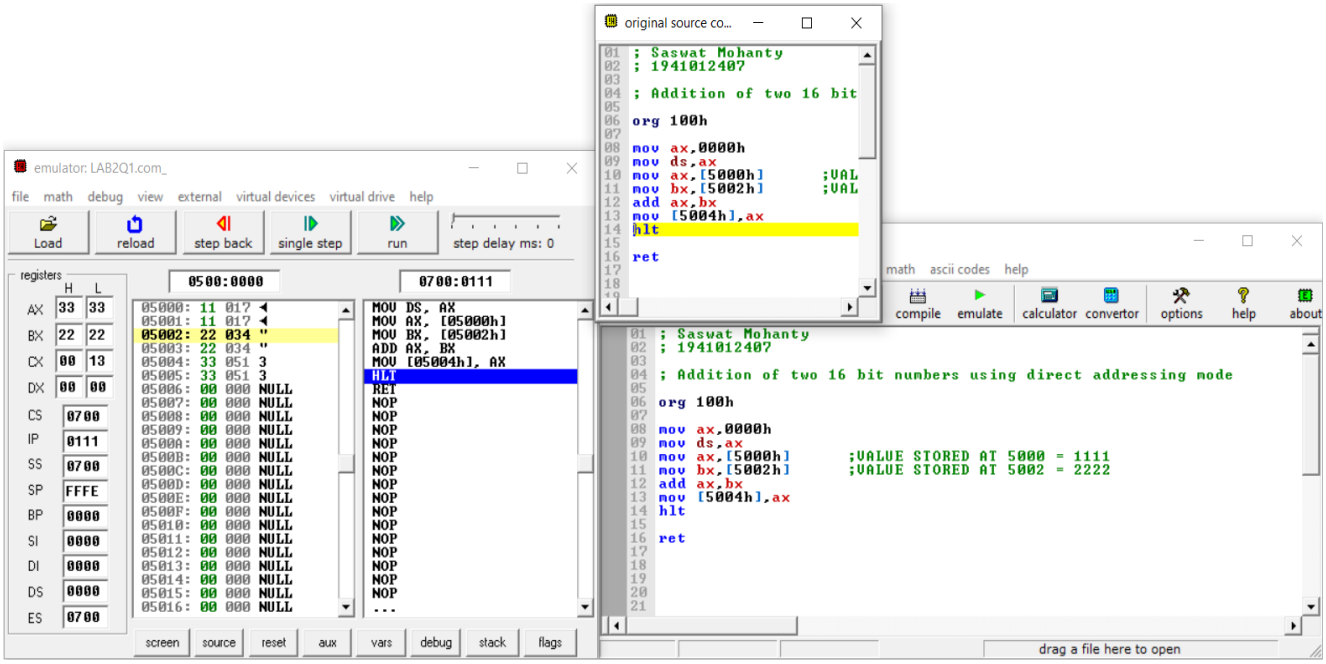
```
; Saswat Mohanty
; 1941012407

; Multiplication of two 16 bit numbers using direct addressing mode

org 100h

mov ax,0000h
mov ds,ax
mov ax,[3000h]      ;VALUE STORED AT 3000 = 1111
mov bx,[3002h]      ;VALUE STORED AT 3002 = 2222
mul bx
mov [3004h],ax
mov [3006h],dx
hlt

ret
```

For Obj. 4

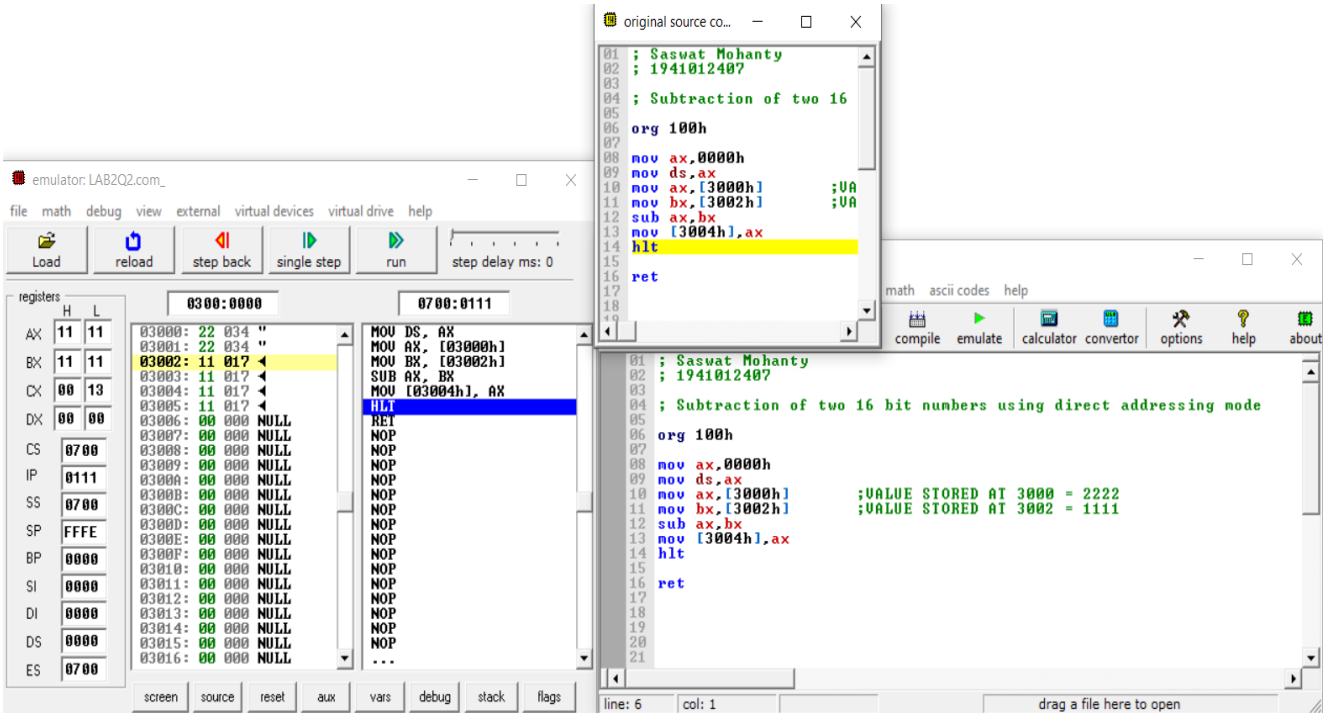
```
; Saswat Mohanty
; 1941012407

; Division of two 16 bit numbers using direct addressing mode

org 100h

mov ax,0000h
mov ds,ax
mov ax,[3000h]      ;VALUE STORED AT 3000 = 6666
mov bx,[3002h]      ;VALUE STORED AT 3002 = 2222
div bx
mov [3004h],ax
mov [3006h],dx
hlt

ret
```

Observations (with screen shots):

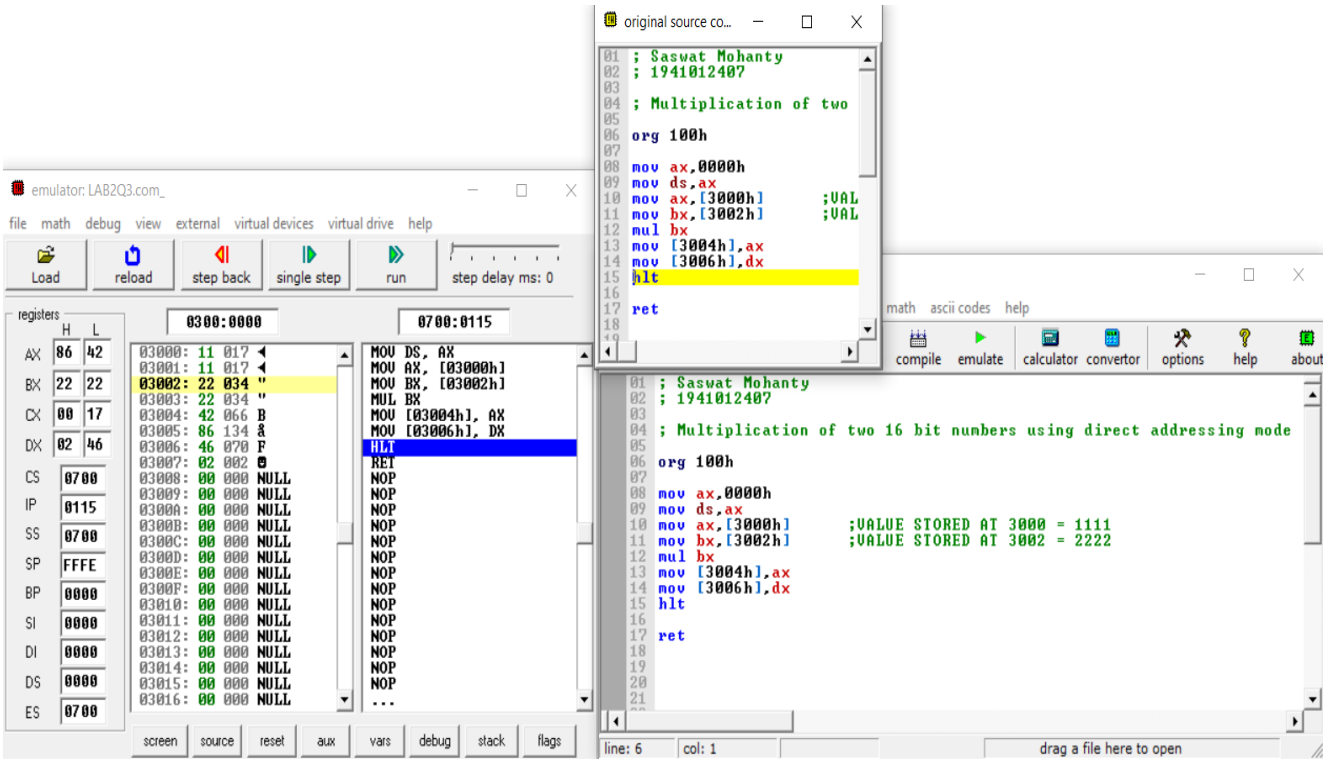
For Obj. 1



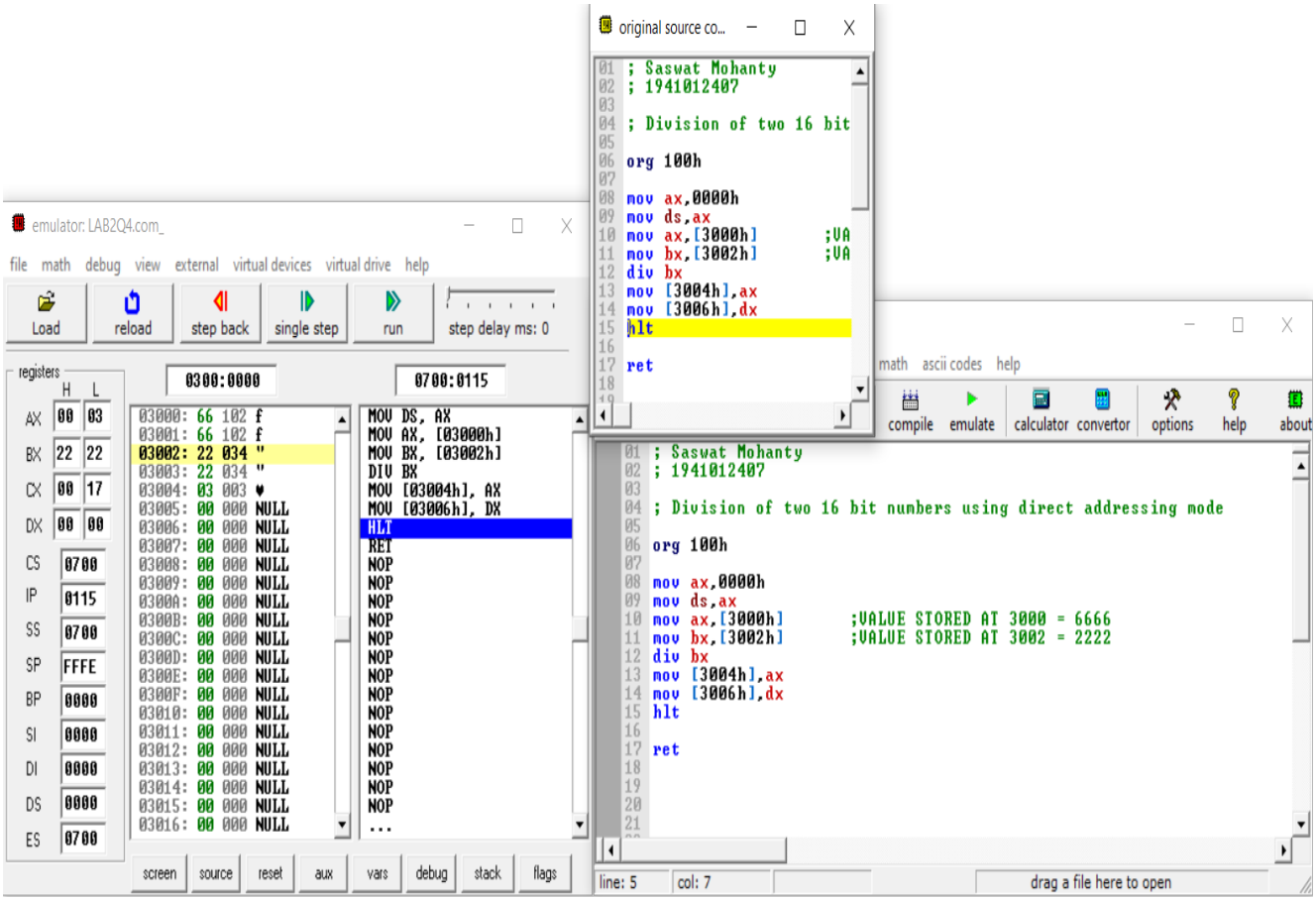
For Obj. 2



For Obj. 3



For Obj. 4



Conclusion:

For Obj. 1:

It can be concluded that the sum of numbers when dry run and executed in system found to be same. Thus, the program to add two 16-bit numbers was executed.

For Obj. 2:

It can be concluded that the difference of numbers when dry run and executed in system found to be same. Thus, the program to subtract two 16-bit numbers was executed.

For Obj. 3:

It can be concluded that the product of numbers when dry run and executed in system found to be same. Thus, the program to multiply two 16-bit numbers was executed.

For Obj. 4:

It can be concluded that the division of numbers when dry run and executed in system found to be same. Thus, the program to divide two 16-bit numbers was executed.

IV. POST LAB:

1. State and explain the different logical instructions of 8086.

Opcode	Operand	Description
AND	D,S	Used for adding each bit in a byte/word with the corresponding bit in another byte/word.
OR	D,S	Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.
NOT	D	Used to invert each bit of a byte or word.
XOR	D,S	Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another

		byte/word.
TEST	D,S	Used to add operands to update flags, without affecting operands.
SHR	D,C	Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.
SHL/SAL	D,C	Used to shift bits of a byte/word towards left and put zero(S) in LSBs.
ROR	D,C	Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].
ROL	D,C	Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].
RCR	D,C	Used to rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.
RCL	D,C	Used to rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

2. Subtract two 16 bit numbers 20H and 06H, and store the difference.

-20

-06

=1A

3. Explain briefly any five arithmetic instructions.

- **ADD** − Used to add the provided byte to byte/word to word.
- **SUB** − Used to subtract the byte from byte/word from word.
- **MUL** − Used to multiply unsigned byte by byte/word by word.
- **DIV** − Used to divide the unsigned word by byte or unsigned double word by word.
- **INC** − Used to increment the provided byte/word by 1.

4. Write the function of the following machine control instructions

a) **WAIT** - Event Wait.

b) **HLT** - Halt CPU.

c) **NOP** - No Operation.

d) **ESC** - Escape.

Computer Organization and Architecture (EET2211)

LAB III: Evaluate Different Logical operations on two 16 bit Data

Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar

Branch: Computer Science and Engineering			**Section:** 'D'
S. No.	**Name**	**Registration No.**	**Signature**
52	Saswat Mohanty	1941012407	Saswat Mohanty

Marks: _____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

6. AND two 16 bit numbers using direct addressing mode.
7. OR two 16 bit numbers using direct addressing mode.
8. NOT of a 16 bit number using direct addressing mode.
9. XOR of two 16 bit numbers using direct addressing mode.

II. PRE-LAB

For Obj. 1:

a) Explain direct addressing mode briefly.

It is 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 AND of two 16 bit numbers.

```
mov ax,[1000h]
mov bx,[1002h]
and ax,bx
```

[1000h] = 1234h

[1002h] = 4321h

Output: 220h

c) Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov ax,[3000h]
```

```
mov bx,[3002h]
and ax,bx
mov [3004h],ax
hlt
ret
```

For Obj. 2:

a) Examine & analyze the output obtained from OR of two 16 bit numbers.

mov ax,[1000h]

mov bx,[1002h]

or ax,bx

[1000h] = 1234h

[1002h] = 4321h

Output: 5335h

b) Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov ax,[3000h]
mov bx,[3002h]
or ax,bx
mov [3004h],ax
hlt
ret
```

For Obj. 3:

a) Examine & analyze the output obtained from NOT of a 16 bit number.

mov ax,1234h

not ax

Output: EDCBh

b) Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov ax,[3000h]
not ax
mov [3002h],ax
hlt
ret
```

For Obj. 4:

a) Examine & analyze the output obtained from XOR of two 16 bit numbers.

mov ax,[1000h]

mov bx,[1002h]

xor ax,bx

[1000h] = 1234h

[1002h] = 4321h

Output: 5115h

b) Write the assembly code.

```
org 100h
```

```
mov ax,0000h
mov ds,ax
mov ax,[3000h]
mov bx,[3002h]
xor ax,bx
mov [3004h],ax
hlt
ret
```

III. LAB:

Assembly Program:

For Obj. 1

```
; SASWAT MOHANTY
; 1941012407

; AND two 16 bit numbers using direct addressing mode

org 100h

mov ax,0000h
mov ds,ax
mov ax,[3000h]    ; Value stored at 3000 = 0202 -> 0000 0010 0000 0010
mov bx,[3002h]    ; Value stored at 3002 = 0202 -> 0000 0010 0000 0010
and ax,bx         ;                              ------------------------------------------------
mov [3004h],ax    ;                                 AND ->  0000 0010 0000 0010 = 0202

hlt

ret
```

For Obj. 2

```
; SASWAT MOHANTY
; 1941012407
```

```
; OR two 16 bit numbers using direct addressing mode

org 100h

mov ax,0000h
mov ds,ax
mov ax,[3000h]    ; Value stored at 3000 = 0202 -> 0000 0010 0000 0010
mov bx,[3002h]    ; Value stored at 3002 = 0303 -> 0000 0011 0000 0011
or ax,bx          ;                              ---------------------------------------------
mov [3004h],ax    ;                                OR -> 0000 0011 0000 0011 = 0303

hlt

ret
```

For Obj. 3

```
; SASWAT MOHANTY
; 1941012407

; NOT of a 16 bit number using direct addressing mode

org 100h

mov ax,0000h
mov ds,ax
mov ax,[3000h]    ; Value stored at 3000 = 0202 -> 0000 0010 0000 0010
not ax            ;                          ---------------------------------------------
mov [3002h],ax    ;                              NOT -> 1111 1101 1111 1101 = FDFD

hlt

ret
```

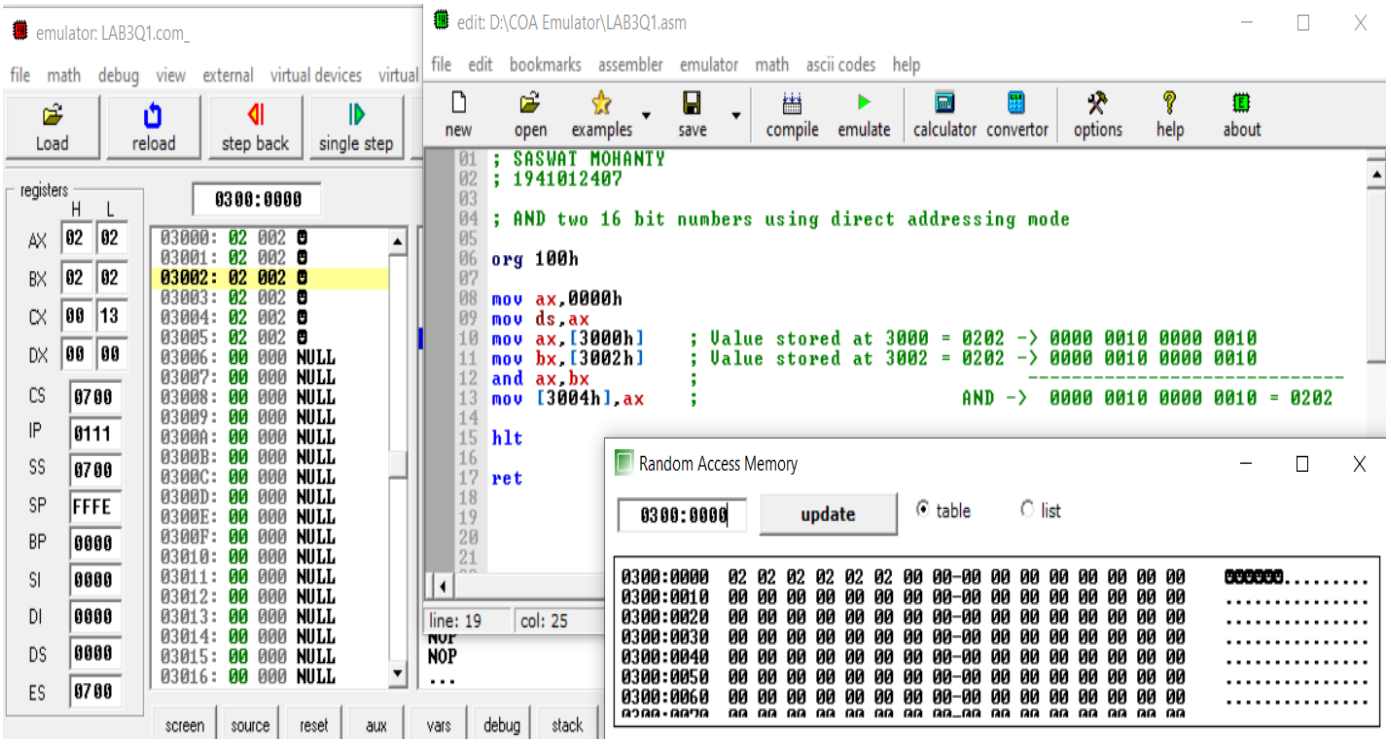
For Obj. 4

```
; SASWAT MOHANTY
; 1941012407

; XOR of two 16 bit numbers using direct addressing mode

org 100h
```

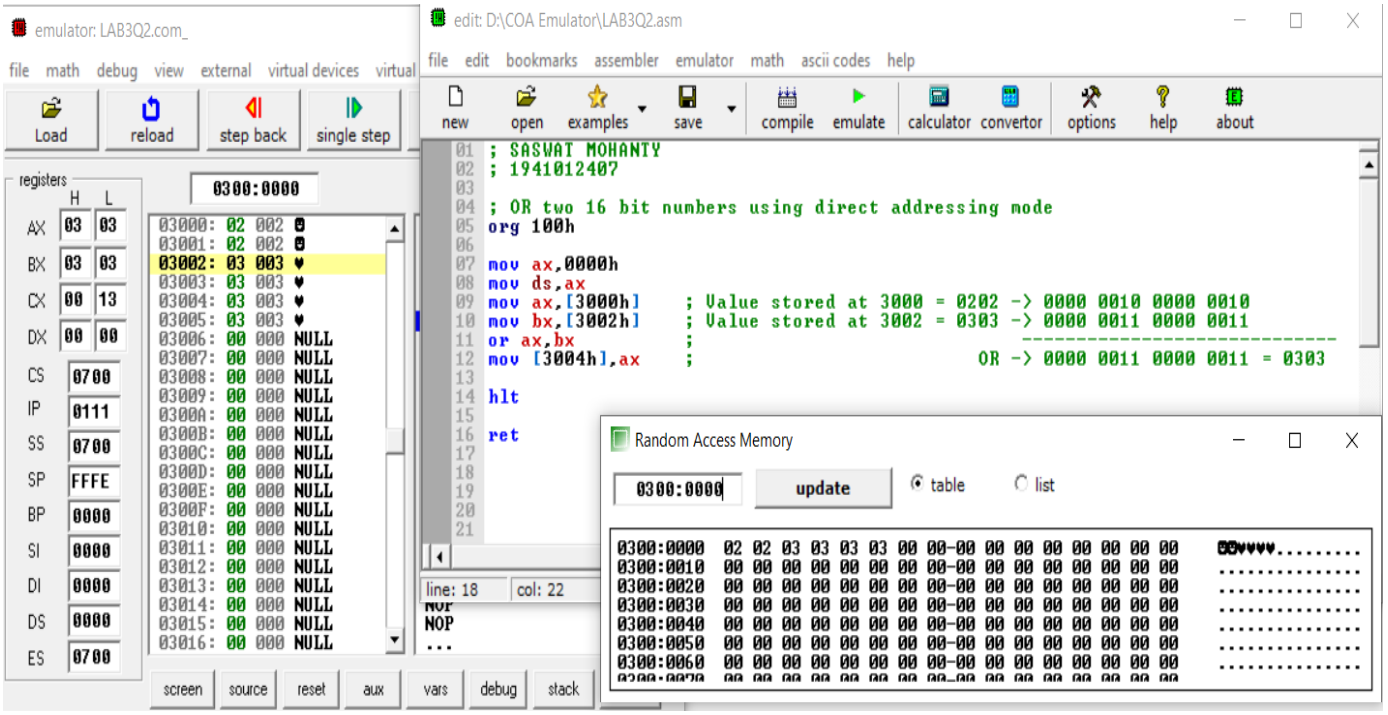
```
mov ax,0000h
mov ds,ax
mov ax,[3000h]   ; Value stored at 3000 = 0202 -> 0000 0010 0000 0010
mov bx,[3002h]   ; Value stored at 3002 = 0303 -> 0000 0011 0000 0011
xor ax,bx        ;                                ---------------------------------------------
mov [3004h],ax   ;                                  XOR -> 0000 0001 0000 0001 = 0101

hlt

ret
```

Observations (with screen shots):

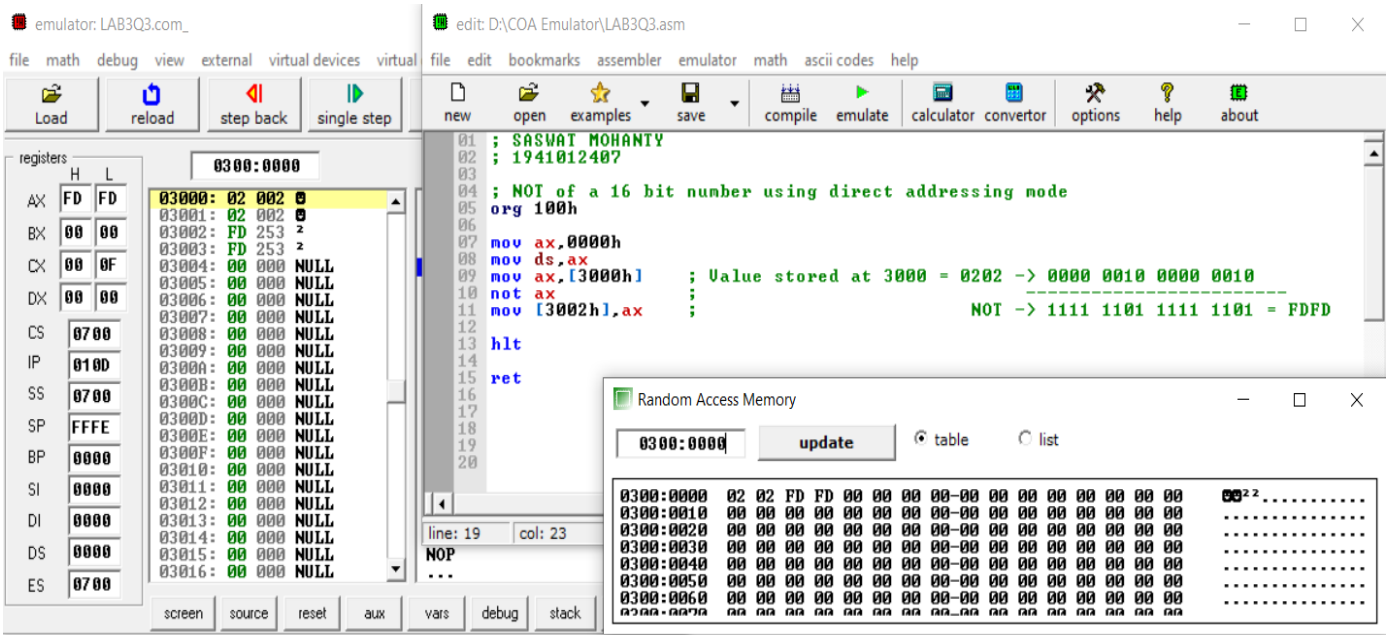
For Obj. 1



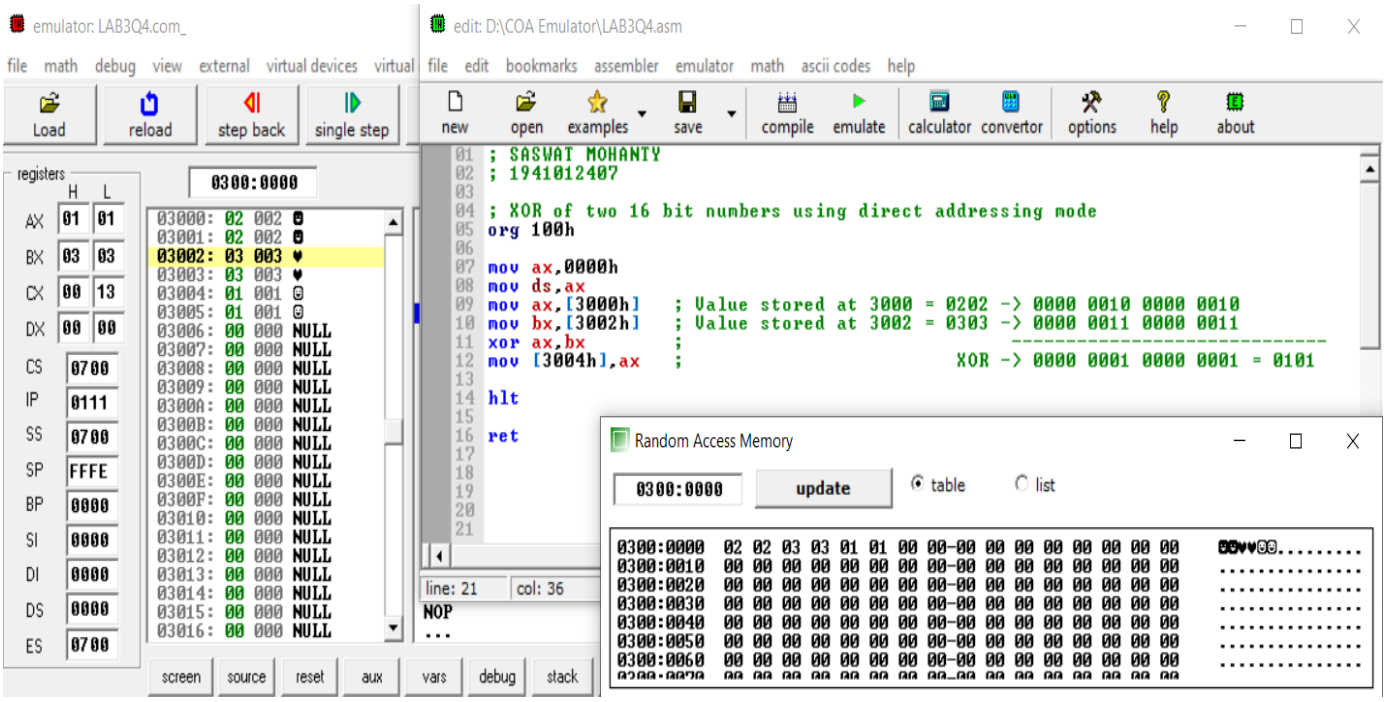
For Obj. 2



For Obj. 3



For Obj. 4



Conclusion:

For Obj. 1:

It can be concluded that the 'and' operation of numbers when dry run and executed in system found to be same. Thus, the program to and two 16-bit numbers was executed.

For Obj. 2:

It can be concluded that the 'or' operation of numbers when dry run and executed in system found to be same. Thus, the program to or two 16-bit numbers was executed.

For Obj. 3:

It can be concluded that the 'not' operation of numbers when dry run and executed in system found to be same. Thus, the program to not a 16-bit number was executed.

For Obj. 4:

It can be concluded that the 'xor' operation of numbers when dry run and executed in system found to be same. Thus, the program to xor two 16-bit numbers was executed.

IV. POST LAB:

1. Enlist the advantages of assembly language programming over machine language.

- It allows complex jobs to run in a simpler way.
- It is memory efficient, as it requires less memory.
- It is faster in speed, as its execution time is less.
- It is mainly hardware-oriented.
- It requires less instruction to get the result.
- It is used for critical jobs.
- It is not required to keep track of memory locations.
- It is a low-level embedded system.

2. Write the function of the following arithmetic instructions

a) ADC b) INC c) DEC d) SBB e) DAA

a) **ADC: -** Used to add with carry.

b) **INC: -** Used to increment the provided byte/word by 1.

c) **DEC: -** Used to decrement the provided byte/word by 1.

d) **SBB: -** Used to perform subtraction with borrow.

e) **DAA: -** Used to adjust the decimal after the addition/subtraction operation.

3. Write the function of the following logical instructions

b) SHL/SAL b) SHR c) SAR d) ROR e) ROL

a) **SHL/SAL: -** Used to shift bits of a byte/word towards left and put zero(S) in LSBs.

b) **SHR: -** Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.

c) **SAR: -** Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB.

d) **ROR: -** Used to rotate bits of byte/word towards the right, i.e., LSB to MSB and to Carry Flag [CF].

e) **ROL: -** Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].

Computer Organization and Architecture (EET2211)

LAB IV: Product and Division of Two Numbers without using Arithmetic Instructions

Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar

Branch: Computer Science and Engineering			**Section:** 'D'
S. No.	**Name**	**Registration No.**	**Signature**
52	Saswat Mohanty	1941012407	Saswat Mohanty

Marks: _____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1. Multiply two 16 bit numbers without using arithmetic instructions.
2. Divide two 16 bit numbers without using arithmetic instructions.

II. PRE-LAB

For Obj. 1:

a) Find the product and quotients of two 16 bit numbers.

Let the two 16bit numbers be 32(0020h) and 8(03h). Their product is 256 (0100h) and quotient is 3.

b) Write the assembly code.

```
org 100h
mov ax, 0000h
mov ds, ax
mov ax,[3000h]
mov cl, 02h
sal ax, cl
mov [3002h], ax
hlt
ret
```

For Obj. 2:

a) Find the quotient and remainder obtained from division of two 16 bit numbers.

Let two number be 200(00c8h) and 4(0004h). Quotient is 50 (00032h) and remainder is 0.

b) Write the assembly code.

```
org 100h
mov ax, 0000h
mov ds, ax
mov ax,[3000h]
mov cl,02h
shr ax,cl
mov [3002h], ax
hlt
ret
```

III. LAB:

Assembly Program:

For Obj. 1:

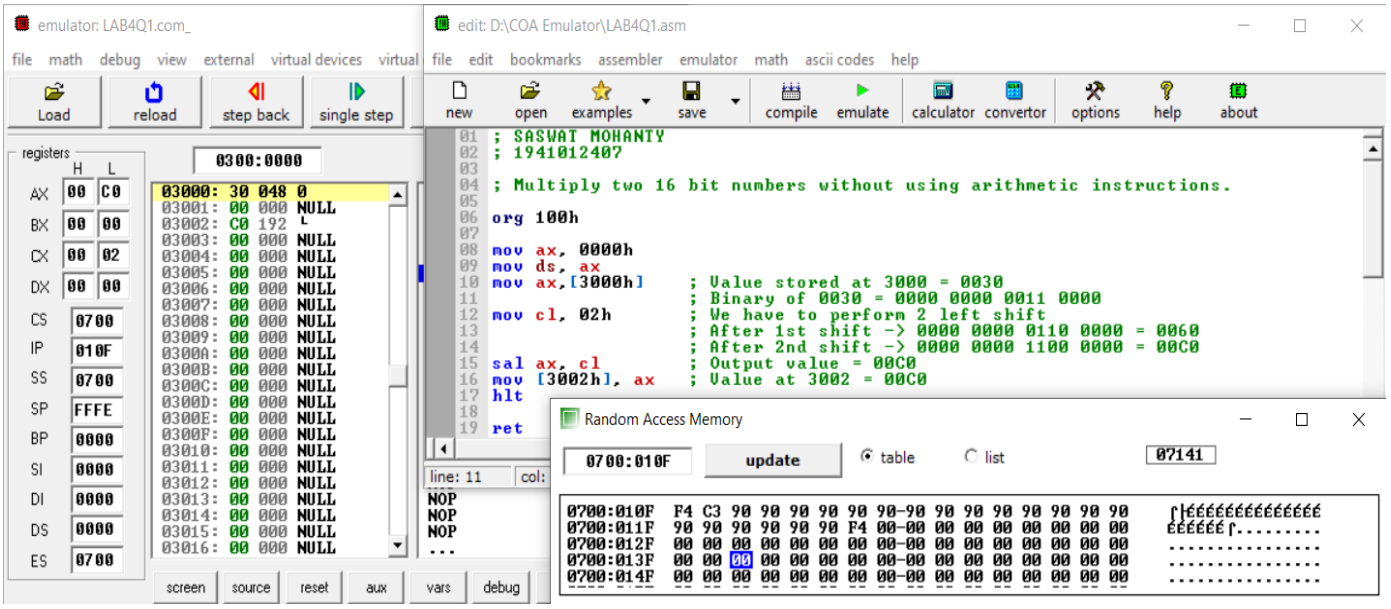
```
; SASWAT MOHANTY
; 1941012407

; Multiply two 16 bit numbers without using arithmetic
instructions.

org 100h

mov ax, 0000h
mov ds, ax
mov ax,[3000h]    ; Value stored at 3000 = 0030
                  ; Binary of 0030 = 0000 0000 0011 0000
mov cl, 02h       ; We have to perform 2 left shift
                  ; After 1st shift -> 0000 0000 0110 0000 = 0060
                  ; After 2nd shift -> 0000 0000 1100 0000 = 00C0
sal ax, cl        ; Output value = 00C0
mov [3002h], ax   ; Value at 3002 = 00C0
hlt

ret
```

For Obj. 2:

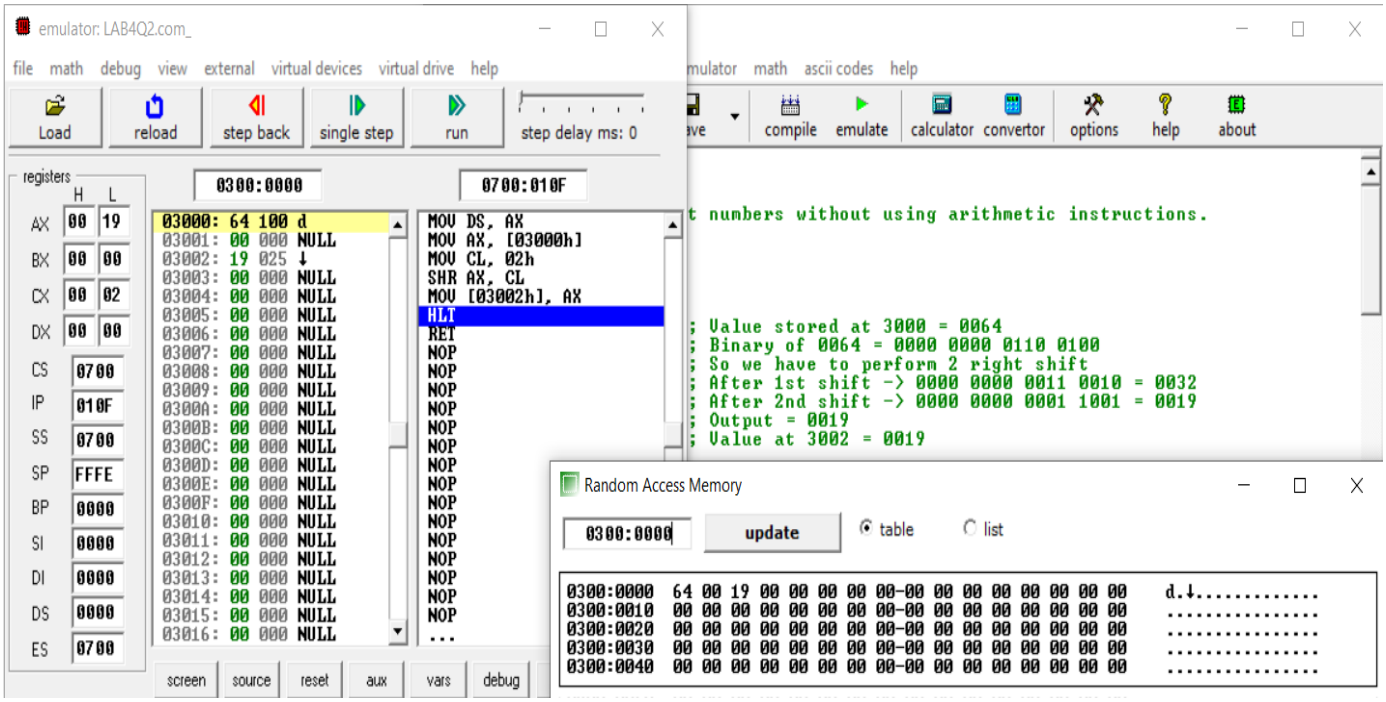
```
; SASWAT MOHANTY
; 1941012407

; Divide  two 16 bit numbers without using arithmetic instructions.

org 100h

mov ax, 0000h
mov ds, ax
mov ax,[3000h]    ; Value stored at 3000 = 0064
                  ; Binary of 0064 = 0000 0000 0110 0100
mov cl,02h        ; So we have to perform 2 right shift
                  ; After 1st shift -> 0000 0000 0011 0010 = 0032
                  ; After 2nd shift -> 0000 0000 0001 1001 = 0019
shr ax,cl         ; Output = 0019
mov [3002h], ax   ; Value at 3002 = 0019
hlt

ret
```

Observations (with screen shots):

For Obj. 1:



For Obj. 2:



Conclusion:

From the above experiment we conclude that the given objective i.e. multiplication and division of any number can be done, provided the divisor is a multiple of 2, by shifting the number left or right 'X' times where x is the power to 2 which when calculated gives the divisor.

IV. POST LAB:

1. **Briefly discuss the instructions used in objectives 1.**

org 100h

mov ax, 0000h

mov ds, ax

mov ax,[3000h] *// at 3000 memory location we store the multiplicand*

(In our case 0030h i.e. 48)

```
mov cl, 02h          // here we store the multiplier (In our case 02h i.e. 4)
sal ax, cl           // then we left shift the multiplicand 2 places to obtain
                        the output (In our case 00C0h i.e. 192.)
mov [3002h], ax      // result stored in ax is then shifted to 3002 memory
                        location.

hlt
ret
```

2. Briefly discuss the instructions used in objectives 2.

```
org 100h
mov ax, 0000h
mov ds, ax
mov ax,[3000h]       // at 3000 memory location we store the divisor
                        (In our case 0064h i.e. 100)
mov cl,02h           // here we store the dividend (In our case 02h i.e. 4)
shr ax,cl            // then we right shift the divisor 2 places to obtain
                        the output (In our case 0019h i.e. 25)
mov [3002h], ax      // result stored in ax is then shifted to 3002 memory
                        location.

hlt
ret
```

3. What is the difference between the microprocessor and microcontroller?

Microprocessor	Microcontroller
Microprocessor consists of only a Central Processing Unit.	Micro Controller contains a CPU, Memory, I/O all integrated into one chip.
Microprocessor is used in Personal Computers.	Micro Controller is used in an embedded system.

Microprocessor uses an external bus to interface to RAM, ROM, and other peripherals.	Microcontroller uses an internal controlling bus.
Microprocessors are based on Von Neumann model	Micro controllers are based on Harvard architecture.
Microprocessor is complicated and expensive, with a large number of instructions to process	Microcontroller is inexpensive and straightforward with fewer instructions to process.

4. What is assembler?

An assembler is a program that converts assembly language into machine code. It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by a specific type of processor.

Computer Organization and Architecture (EET2211)

LAB V: Addition of two BCD numbers

Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar

Branch: Computer Science and Engineering			**Section:** 'D'
S. No.	**Name**	**Registration No.**	**Signature**
52	Saswat Mohanty	1941012407	Saswat Mohanty

Marks: _____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1) Write a program to find the sum of two BCD numbers.

II. PRE-LAB

For Obj. 1:

a) Find the sum of two BCD numbers.

[1000h] = 2222h

[1002h] = 1111h

Output: 3333h

b) Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov al,[3000h]
mov bl,[3002h]
add al,bl
daa
mov [3004h],al
mov al,[3001h]
mov bl,[3003h]
adc al,bl
daa
mov [3005h],al
mov al,00
adc al,al
mov [3006h],al
hlt
ret
```

III. LAB:

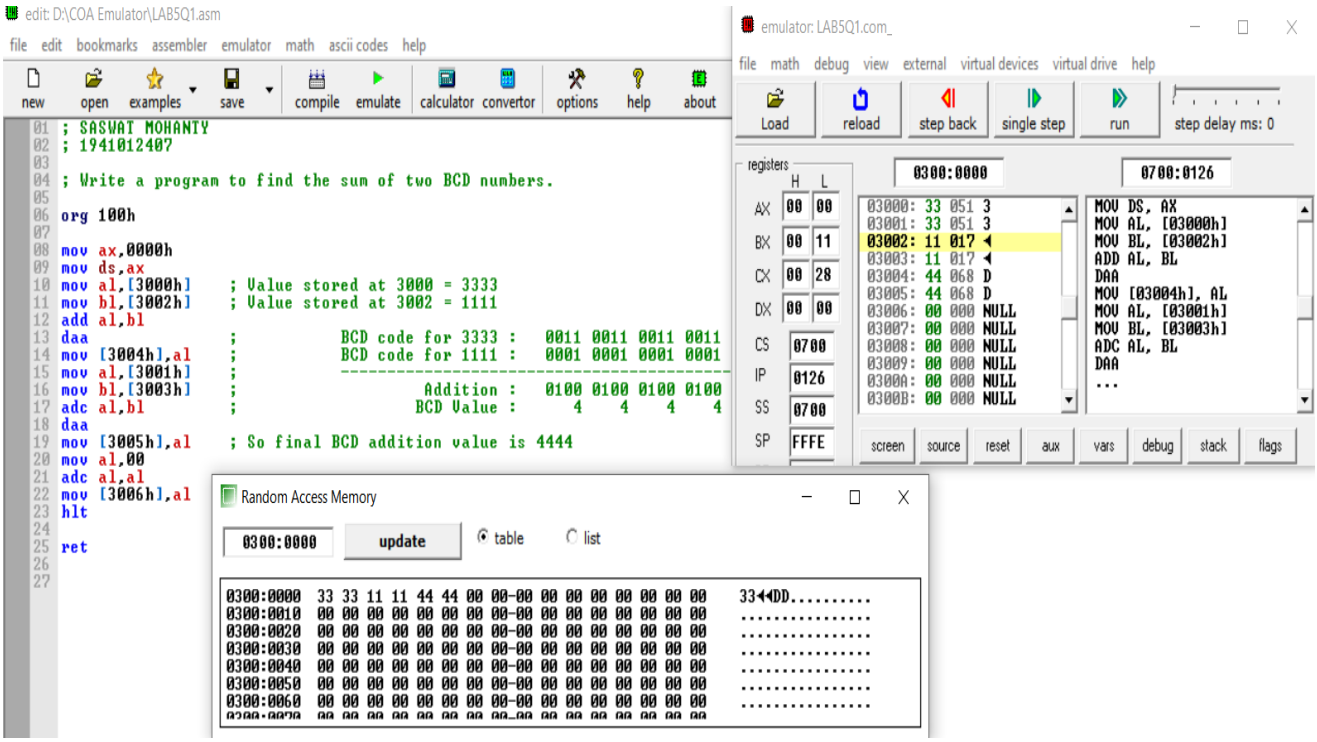
Assembly Program:

For Obj. 1:

```
; SASWAT MOHANTY
; 1941012407

; Write a program to find the sum of two BCD numbers.

org 100h

mov ax,0000h
mov ds,ax
mov al,[3000h]    ; Value stored at 3000 = 3333
mov bl,[3002h]    ; Value stored at 3002 = 1111
add al,bl
daa               ;   BCD code for 3333 :   0011 0011 0011 0011
mov [3004h],al    ;   BCD code for 1111 :   0001 0001 0001 0001
mov al,[3001h]    ;                         -----------------------------------
mov bl,[3003h]    ;              Addition :   0100 0100 0100 0100
adc al,bl         ;             BCD Value :      4    4    4    4
daa
mov [3005h],al    ; So final BCD addition value is 4444
mov al,00
adc al,al
mov [3006h],al
hlt

ret
```

Observations (with screen shots):

For Obj. 1:



Conclusion:

It can be concluded that the addition of two BCD numbers when dry run and executed in system found to be same. Thus, the program to find the sum of two BCD numbers was executed.

IV. POST LAB:

1. **What is the maximum memory size that can be addressed by 8086?**

   In 8086 microprocessor the total memory addressing capability is 1MB. For representing 1MB there are minimum 4 hex digits are required i.e., 20 bits. 8086 microprocessors have fourteen 16 bit registers (i.e. there are no registers for representing 20 bit address). So, the total memory can be divided into 16 separate logical segments and each segment capacity is 64KB (i.e., 16 * 64 KB = 1MB).

2. Which of the following is not a data copy/transfer instruction? Explain.

a) MOV b) PUSH

c) DAS d) POP

DAS is the answer because, it's used to adjust decimal after subtraction.

3. Write Down the Comparisons between the 8086 and 8088?

	8086	**8088**
Clock Speeds	5MHz, 8MHz, 10MHz	5MHz, 8MHz
Bus Width	16 bits	8 bits
Number of Transistors	29,000	29,000
Feature size	3	6
Addressable Memory	1MB	1MB

Computer Organization and Architecture (EET2211)

LAB VI: Find 1's and 2's complement of a number

Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar

Branch: Computer Science and Engineering			**Section:** 'D'
S. No.	**Name**	**Registration No.**	**Signature**
52	Saswat Mohanty	1941012407	Saswat Mohanty

Marks: _____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1. Write a program to find 1's and 2's complement of a given number without using logical instructions.

II. PRE-LAB

For Obj. 1:

a) Find 1's and 2's complement of a given number.

[3000h] = 2222h

Output: 1s Complement = EEEEh

2s Complement = EEEFh

b) Write the assembly code.

```
org 100h
mov ax,0000h
mov ds, ax
mov ax,[3000h]
mov bx,[3002h]
sub ax,bx
mov [3004h],ax
inc ax
mov [3006h],ax
hlt
ret
```

III. LAB:

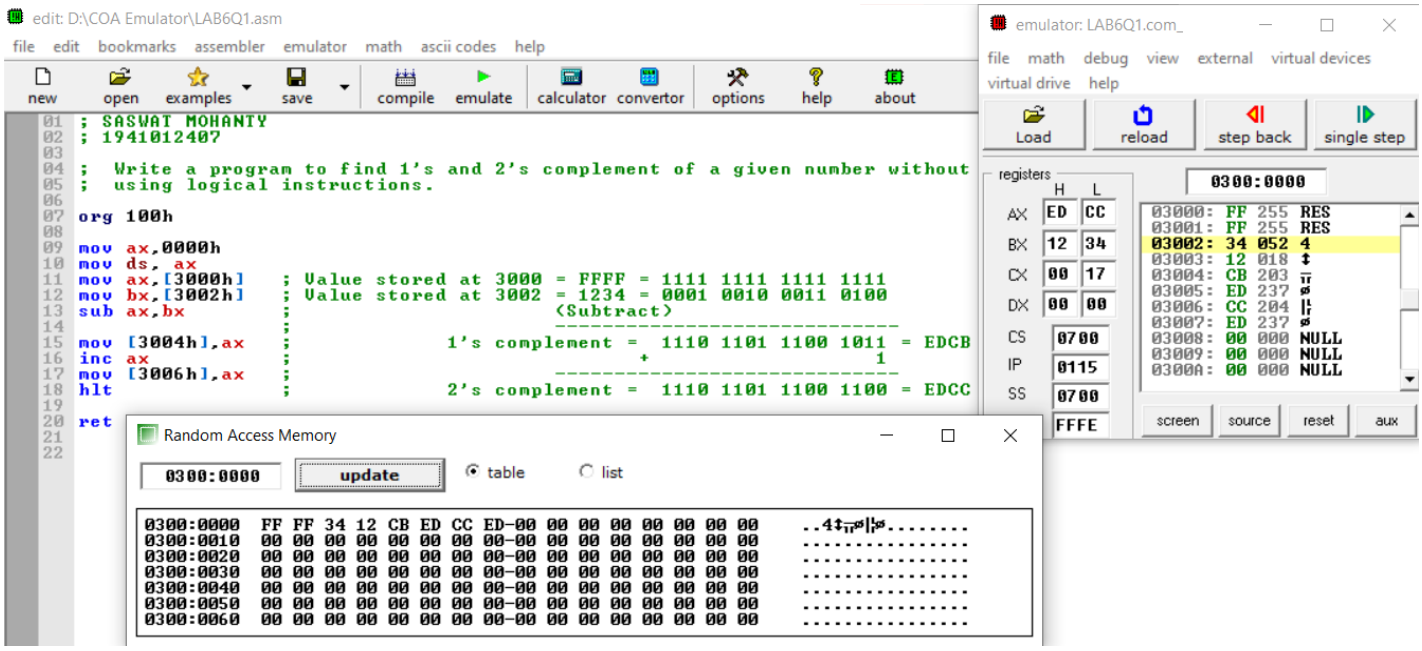
Assembly Program:

For Obj. 1:

```
; SASWAT MOHANTY
; 1941012407

;  Write a program to find 1's and 2's complement of a given number without
;  using logical instructions.

org 100h

mov ax,0000h
mov ds, ax
mov ax,[3000h]   ; Value stored at 3000 = FFFF = 1111 1111 1111 1111
mov bx,[3002h]   ; Value stored at 3002 = 1234 = 0001 0010 0011 0100
sub ax,bx        ;                            (Subtract)
                 ;                            ---------------------------------------------
mov [3004h],ax   ;              1's complement =  1110 1101 1100 1011 = EDCB
inc ax           ;                                    +                    1
mov [3006h],ax   ;                            ---------------------------------------------
hlt              ;              2's complement =  1110 1101 1100 1100 = EDCC

ret
```

Observations (with screen shots):

For Obj. 1:



```
; SASWAT MOHANTY
; 1941012407

;  Write a program to find 1's and 2's complement of a given number without
;  using logical instructions.

org 100h

mov ax,0000h
mov ds, ax
mov ax,[3000h]   ; Value stored at 3000 = FFFF = 1111 1111 1111 1111
mov bx,[3002h]   ; Value stored at 3002 = 1234 = 0001 0010 0011 0100
sub ax,bx        ;                            (Subtract)
                 ;                            ---------------------------------------------
mov [3004h],ax   ;              1's complement =  1110 1101 1100 1011 = EDCB
inc ax           ;                                    +                    1
mov [3006h],ax   ;                            ---------------------------------------------
hlt              ;              2's complement =  1110 1101 1100 1100 = EDCC

ret
```

Conclusion:

It can be concluded that 1's and 2's complement of a given number when dry run and executed in system found to be same. Thus, the program to find the 1's and 2's complement of a given number was executed.

IV. POST LAB:

1. Briefly explain the logical address, base segment address and physical address?

Logical address is contained in the 16-bit IP, BP, SP, BX, SI or DI. It is also known as the offset address or the effective address.

The base segment address is contained in one of the 16bit contents of the segment registers CS, DS, ES, SS.

The physical address or the real address is formed by combining the offset and base segment addresses. This address is 20bit and is primarily used for the accessing of the memory.

2. Differentiate between CISC and RISC.

CISC	RISC
Stands for Complex Instruction Set Computers	Stands for reduced Instruction Set Computers
A full set of computer instructions that intends to provide the necessary capabilities in an efficient way	An instruction set architecture that is designed to perform a smaller number of computer instructions so that it can operate at a higher speed
Instruction cycles can take several clock cycles to execute	Single cycle instructions execution takes place
Hardware centric design	Software centric design

3. Explain briefly the advantages of pipelining in 8086.

Advantages of pipelining:

The EU always reads the next instruction byte from the queue in BIU. This is much faster than sending out an address to the memory and waiting for the next instruction byte to come.

In short pipelining eliminates the waiting time of EU and speeds up the processing.

The 8086 BIU will not initiate a fetch unless and until there are two empty bytes in its queue.

4. Briefly explain the following:

a) Stack Pointer (SP) **b) Base Pointer (BP)**

c) Destination Index (DI) **d) Source Index (SI)**

a) **Stack Pointer (SP): -** The Stack Pointer (SP) register is used to indicate the location of the last item put onto the stack.

b) **Base Pointer (BP): -** The base pointer refers to the bottom of the stack, which normally refers to higher addresses as it grows towards lower.

c) **Destination Index (DI): -** The Destination Index register used as a pointer to the current character being written or compared in a string instruction.

d) **Source Index (SI): -** The Source Index register is used as source index for string operations.

Computer Organization and Architecture (EET2211)

LAB VII: Swap the upper nibble of a word with the lower nibble content of an accumulator

Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar

Branch: Computer Science and Engineering			**Section:** 'D'
S. No.	**Name**	**Registration No.**	**Signature**
52	Saswat Mohanty	1941012407	Saswat Mohanty

Marks: _____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1) Write a program to swap the upper nibble of a word with the lower nibble content of an accumulator.

II. PRE-LAB

For Obj. 1:

a) Swap the upper nibble of a word with the lower nibble content of an accumulator.

[5000h] = 1234h
Output: 3412h

b) Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov ax,[5000h]
mov cl,08h
rol ax,cl
mov [5002h],ax
hlt
ret
```

III. LAB:

Assembly Program:

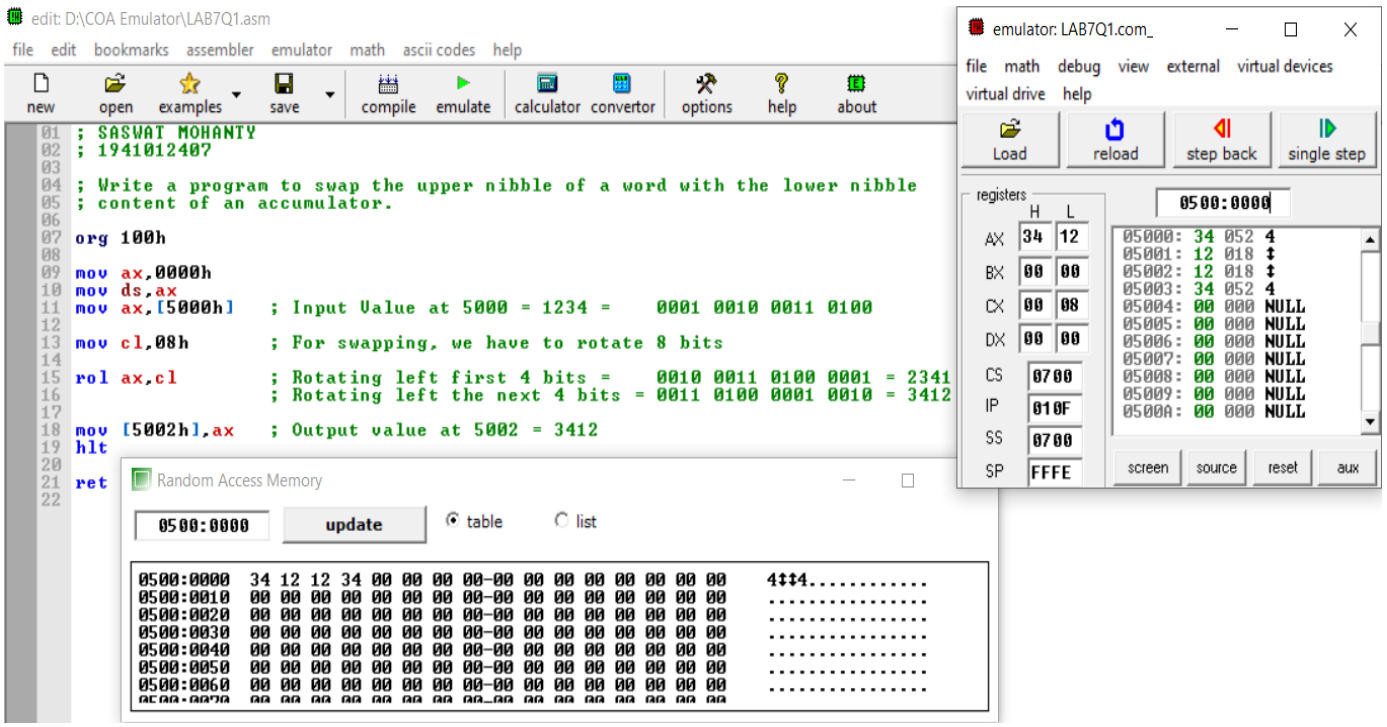
For Obj. 1:

```
; SASWAT MOHANTY
; 1941012407
```

```
; Write a program to swap the upper nibble of a word with the lower nibble
; content of an accumulator.

org 100h

mov ax,0000h
mov ds,ax
mov ax,[5000h]    ; Input Value at 5000 = 1234 =    0001 0010 0011 0100

mov cl,08h        ; For swapping, we have to rotate 8 bits

rol ax,cl         ; Rotating left first 4 bits =    0010 0011 0100 0001 = 2341
                  ; Rotating left the next 4 bits = 0011 0100 0001 0010 = 3412

mov [5002h],ax    ; Output value at 5002 = 3412
hlt

ret
```

Observations (with screen shots):

For Obj. 1:



Conclusion:

It can be concluded that swap the upper nibble of a word with the lower nibble content of an accumulator when dry run and executed in system found to be same. Thus, the program to swap the nibbles was executed.

IV. POST LAB:

1. Explain briefly the advantages of memory segmentation in 8086.

Advantages of memory segmentation in 8086:-

- It allows to processes to easily share data.
- 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.

2. Explain the IAS instruction format.

The IAS machine was a binary computer with a 40-bit word, storing two 20-bit instructions in each word. The memory was 1,024 words (5.1 kilobytes). Negative numbers were represented in two's complement format. It had two general-purpose registers available: the Accumulator (AC) and Multiplier/Quotient (MQ).

3. Briefly explain the following flags of 8086:

b) Carry Flag (CF) **b) Parity Flag (PF)** **c) Adjust Flag (AF)**

d) Zero Flag (ZF) **e) Sign Flag (SF)** **f) Overflow Flag (OF)**

a) **Carry Flag (CF): -** Holds the carry after addition or borrow after subtraction. Also indicates some error conditions as dictated by some programs and procedures.

b) **Parity Flag (PF): -** PF=0= odd parity; PF=1=even parity

c) **Adjust Flag (AF): -** Holds the carry (half carry) after addition or borrow after subtraction between bit positions 3 and 4 of the result (e.g. in BCD addition or subtraction)

d) **Zero Flag (ZF): -** Shows the result of the arithmetic or logic operation.

e) **Sign Flag (SF): -** Holds the sign of the result after an arithmetic/logic instruction execution.

f) **Overflow Flag (OF): -** Overflow occurs when signed numbers are added or subtracted. An overflow indicates the result has exceeded the capacity of the machine.

Computer Organization and Architecture (EET2211)

LAB VIII: Calculate average of N 16-bit numbers

Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar

Branch:			**Section:**
S. No.	**Name**	**Registration No.**	**Signature**
52	Saswat Mohanty	1941012407	Saswat Mohanty

Marks: _____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1. Write a program to calculate average of N 16-bit numbers

II. PRE-LAB

For Obj. 1:

a. Calculate average of N 16-bit numbers.

[1500h] = 03h

[1501h] = 10h

[1502h] = 10h

[1503h] = 10h

Output: 10h

b. Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov si,1500h
mov di,1510h
mov ax,0000h
mov cl,[si]
mov bl,cl
inc si
loop: add al,[si]
      adc ah,00
      inc si
      dec cl
      jnz loop
div bl
```

```
mov [di],ax
hlt
ret
```

III. LAB:

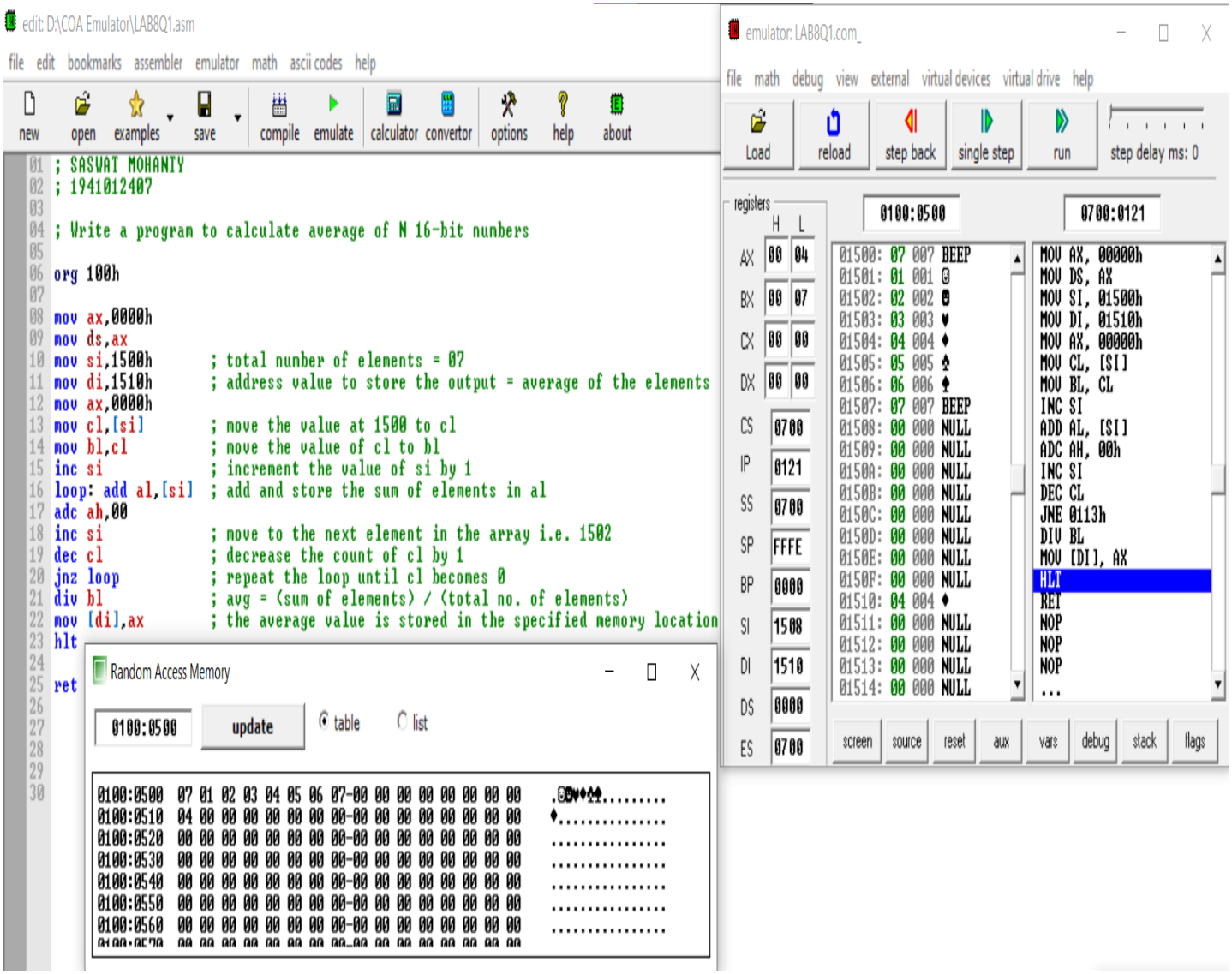
Assembly Program:

For Obj. 1:

```
; SASWAT MOHANTY
; 1941012407

; Write a program to calculate average of N 16-bit numbers

org 100h

mov ax,0000h
mov ds,ax
mov si,1500h       ; total number of elements = 07
mov di,1510h       ; address value to store the output = average of the elements
mov ax,0000h
mov cl,[si]        ; move the value at 1500 to cl
mov bl,cl          ; move the value of cl to bl
inc si             ; increment the value of si by 1
loop: add al,[si]  ; add and store the sum of elements in al
adc ah,00
inc si             ; move to the next element in the array i.e. 1502
dec cl             ; decrease the count of cl by 1
jnz loop           ; repeat the loop until cl becomes 0
div bl             ; avg = (sum of elements) / (total no. of elements)
mov [di],ax        ; the average value is stored in the specified memory location
hlt

ret
```

Observations (with screen shots):

For Obj. 1:



Conclusion:

It can be concluded to determine the largest number in an array when dry run and executed in system found to be same. Thus, the program to determine the largest number in an array was executed.

IV. POST LAB:

1. What is the maximum internal clock frequency of 8086?

The maximum internal clock frequency of8086 is 5MHz.

2. List few applications of microprocessor-based system.

The use of microprocessor in toys, entertainment equipment and home applications is making them more entertaining and full of features. The use of microprocessors is more widespread and popular. Now the Microprocessors are used in:

- Calculators
- Accounting system
- Games machine
- Complex Industrial Controllers
- Traffic light Control
- Data acquisition systems

3. Briefly explain the following instructions of 8086:

a) JMP b) JZ c) JNZ d) JC e) JNC

a. **JMP: -** Used to jump to the provided address to proceed to the next instruction.

b. **JZ: -** Used to jump if equal/zero flag ZF = 1

c. **JNZ: -** Used to jump if not equal/zero flag ZF = 0

d. **JC: -** Used to jump if carry flag CF = 1

e. **JNC: -** Used to jump if no carry flag (CF = 0)

Computer Organization and Architecture (EET2211)

LAB IX: Determine the largest and smallest number in an array

Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar

Branch:			**Section:**
S. No.	**Name**	**Registration No.**	**Signature**
52	Saswat Mohanty	1941012407	Saswat Mohanty

Marks: _____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1. Write a program to determine the largest number in an array.
2. Write a program to determine the smallest number in an array.

II. PRE-LAB

For Obj. 1:

a. Determine the largest number in an array.

[1500h] = 03h

[1501h] = 13h

[1502h] = 22h

[1503h] = 11h

Output: 22h

b. Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov si,1500h
mov di,1510h
mov cl,[si]
inc si
mov al,[si]
dec cl
l1: inc si
    mov bl,[si]
    cmp al,bl
    jnc again
    mov al,bl
```

```
again: dec cl
       jnz l1
mov [di],al
hlt
ret
```

For Obj. 2:

a. Determine the smallest number in an array.

[1500h] = 03h

[1501h] = 13h

[1502h] = 22h

[1503h] = 11h

Output: 03h

b. Write the assembly code.

```
org 100h
mov ax,0000h
mov ds,ax
mov si,1500h
mov di,1510h
mov cl,[si]
inc si
mov al,[si]
dec cl
l1: inc si
   mov bl,[si]
   cmp al,bl
   jc again
   mov al,bl
```

```
again: dec cl
       jnz l1
mov [di],al
hlt
ret
```

III. LAB:

Assembly Program:

For Obj. 1:

```
; SASWAT MOHANTY
; 1941012407

; Write a program to determine the largest number in an array.

org 100h

mov ax,0000h
mov ds,ax
mov si,1500h    ; Total number of elements = 03
mov di,1510h    ; Memeory address for storing the output
mov cl,[si]     ; Assign the value at 1500 to cl
inc si          ; increment si i.e. to memory location 1501
mov al,[si]     ; assign the value at 1501 to al
dec cl          ; decrease the value of cl by 1
l1: inc si      ; increment si i.e. to memory location 1502
    mov bl,[si] ; assign the value at 1502 to bl
    cmp al,bl   ; compare al and bl to check which value is greater
    jnc again   ; after comaprision if there is no carry then it will jump to the "again" pointer
    mov al,bl   ; if there is carry then it will assign the value in bl  to al
again: dec cl   ; decrease the cl count by 1
       jnz l1   ; until cl count is zero the program keep on moving to "l1" pointer
mov [di],al     ; move the value in al to di i.e. to 1510 memory location
hlt

ret
```

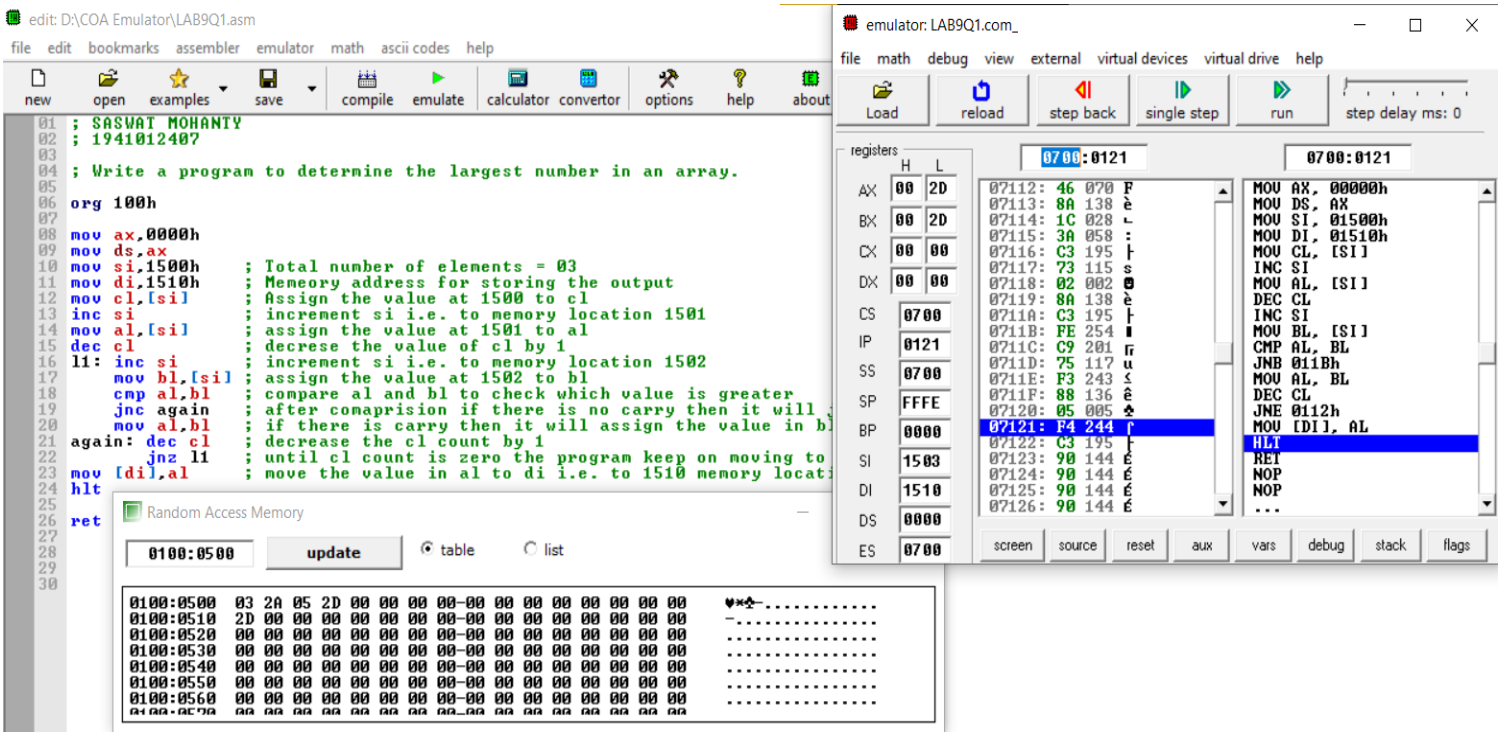
For Obj. 2:

```
; SASWAT MOHANTY
; 1941012407
```

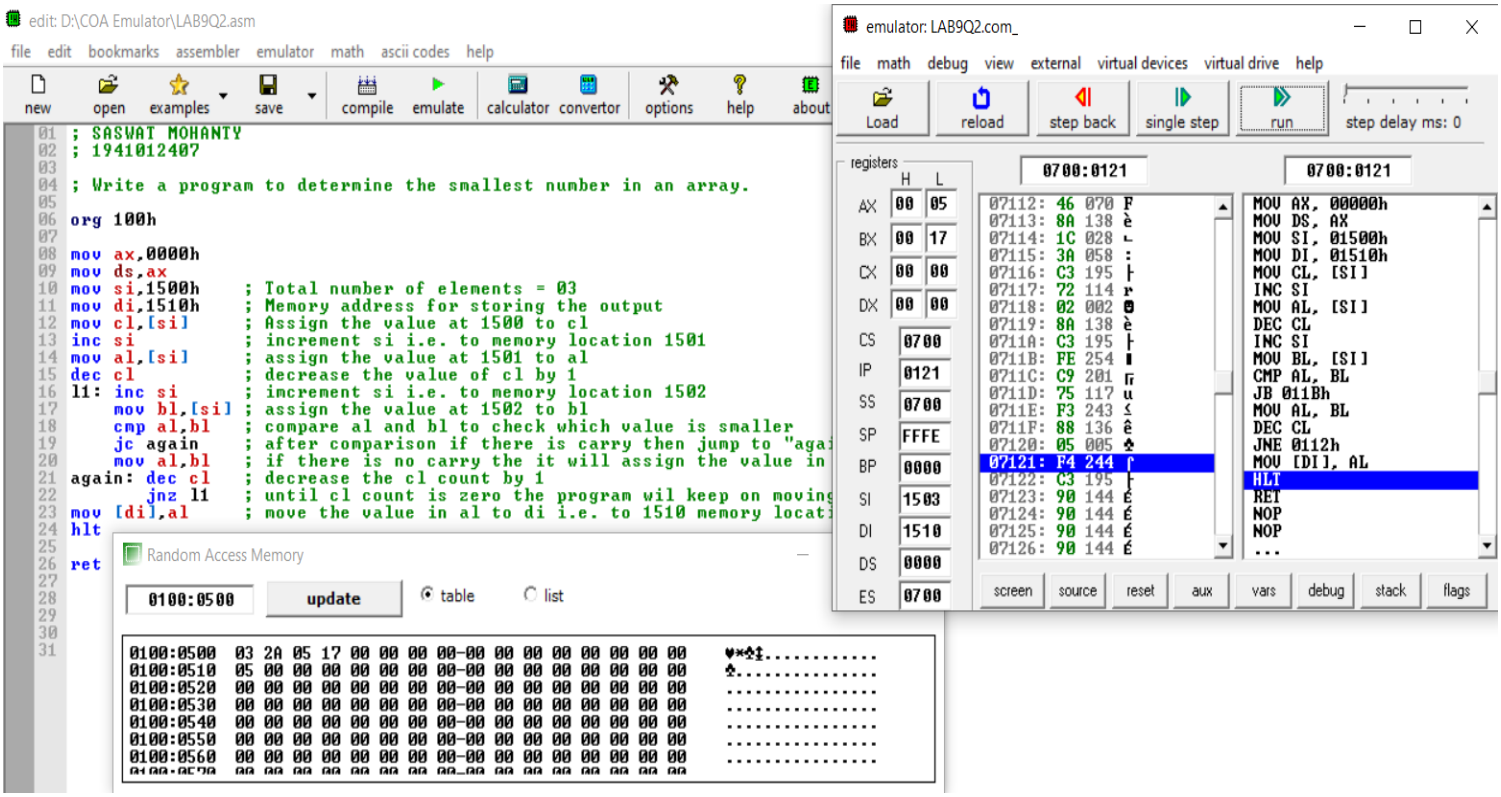
```
; Write a program to determine the smallest number in an array.

org 100h

mov ax,0000h
mov ds,ax
mov si,1500h    ; Total number of elements = 03
mov di,1510h    ; Memory address for storing the output
mov cl,[si]     ; Assign the value at 1500 to cl
inc si          ; increment si i.e. to memory location 1501
mov al,[si]     ; assign the value at 1501 to al
dec cl          ; decrease the value of cl by 1
l1: inc si      ; imcrement si i.e. to memory location 1502
    mov bl,[si] ; assign the value at 1502 to bl
    cmp al,bl   ; compare al and bl to check which value is smaller
    jc again    ; after comparison if there is carry then jump to "again" pointer
    mov al,bl   ; if there is no carry the it will assign the value in bl to al
again: dec cl   ; decrease the cl count by 1
      jnz l1    ; until cl count is zero the program wil keep on moving to "l1" pointer
mov [di],al     ; move the value in al to di i.e. to 1510 memory location
hlt

ret
```

Observations (with screen shots):

For Obj. 1:



For Obj. 2:



Conclusion:

For Obj. 1:

It can be concluded to determine the largest number in an array when dry run and executed in system found to be same. Thus, the program to determine the largest number in an array was executed.

For Obj. 2:

It can be concluded to determine the smallest number in an array when dry run and executed in system found to be same. Thus, the program to determine the smallest number in an array was executed.

IV. POST LAB:

1. What is ARM processor?

An ARM processor is one of a family of CPUs based on the RISC (reduced instruction set computer) architecture developed by Advanced RISC Machines (ARM).

2. Differentiate between ARM processor and RISC.

ARM	RISC
ARM is proprietary.	RISC is open-source.
ARM makes 32-bit and 64-bit RISC multi-core processors.	RISC processors are designed to perform a smaller number of types of computer instructions so that they can operate at a higher speed, performing more millions of instructions per second (MIPS).
ARM has added more complex instructions to increase processor performance (at the expense of higher power consumption).	RISC approach is more successful in reducing overall power consumption, sometimes at the expense of lower performance.

3. Differentiate between ARM processor and 8086.

ARM	8086
Integrated in designs which were manufactured on 28, 16, 14 or 10 nanometer FinFET nodes	Manufactured on a 3-micron process

RICS Design	CISC Design
Consists of a front end, back end (execution engine) and an un-core memory subsystem which includes the L2 cache.	Consists of two main blocks, the BIU and EU

4. Differentiate between ARM processor and microcontroller.

ARM is core for both microprocessor and micro-controller. ARM is based on CPU architecture so we generally call it has microprocessor when placed on a chip if ARM is combined with memories (RAM and ROM) on a single chip we can call it has micro-controller it has limited memory but when coming to microprocessor RAM and ROM are connected externally speed will be more.

5. List few applications of ARM processor-based system.

- ARM processor features include:
- Load/store architecture.
- An orthogonal instruction set.
- Mostly single-cycle execution.
- Enhanced power-saving design.
- 64 and 32-bit execution states for scalable high performance.
- Hardware virtualization support.