

# **MICROPROCESSORS BASED SYSTEMS DESIGN (UCS617)**

## **LAB ASSIGNMENT**

### **Submitted By:**

Gurjot Singh	102203582
Anil Kumar	102383026
Simarpreet Singh	102203606
Samiksha Kak	102203587
Archita Jain	102203613

**Group: 3C34**

**BE Third Year**

### **Submitted To:**

**Dr. Anju Bala**  
(Associate Professor)



**Computer Science and Engineering Department**

**Thapar Institute of Engineering and Technology**

**Patiala – 147001**

**Jan-May 2025**

## **TABLE OF CONTENTS**

<b>Sr. No.</b>	<b>Description</b>	<b>Page No.</b>
	<b>8085 Microprocessor</b>	
1.	Introduction of 8085-microprocessor kit and steps for execution on the kit.	4
2.	Familiarity with 8085-microprocessor kit.	
i)	Write a program to store 8-bit data into one register and then copy that to all registers.	13
ii)	Write a program for addition of two 8-bit numbers	15
iii)	Write a program to add 8-bit numbers using direct and indirect addressing mode	17
iv)	Write a program to add 16-bit numbers using direct and indirect addressing mode	19
v)	Write a program to 8-bit numbers using carry. (using JNC instruction).	23
vi)	Write a program to find 1's complement and 2's complement of 8-bit number.	26
3.	Write a program for the sum of series of numbers.	30
4.	Write a program for data transfer from memory block B1 to memory block B2.	32
5.	Write a program for multiply two 8-bit numbers.	35
6.	Write a program to add ten 8-bit numbers. Assume the numbers are stored in 8500-8509. Store the result in 850A and 850B memory address.	37
7.	Write a program to find the negative numbers in a block of data.	39
8.	Write a program to count the number of one's in a number.	41
9.	Write a program to arrange numbers in Ascending order.	43
10.	Calculate the sum of series of even numbers.	46
11.	Write an assembly language program to verify how many bytes are present in a given set, which resembles 10101101 in 8085	48
12.	Write an assembly language program to find the numbers of even parity in ten consecutive memory locations in 8085.	50
13.	Write an assembly language program to convert a BCD number into its equivalent binary in 8085.	52
14.	Write an assembly language program for exchange the contents of memory location.	54
15.	Write a program to find the largest number in an array of 10 elements.	56

<b>Sr. No.</b>	<b>Description</b>	<b>Page No.</b>
	<b>8086 Microprocessor</b>	
16.	Write an assembly language program to add two 16-bit numbers in 8086.	58
17.	Write an assembly language program to subtract two 16-bit numbers in 8086.	59
18.	Write an assembly language program to multiply two 16-bit numbers in 8086.	60
19.	Write an assembly language program to divide two 16-bit numbers in 8086.	61
20.	Write an assembly language program to demonstrate AAA, AAS, AAM, AAD, DAA and DAS in 8086.	62
21.	Write an assembly language program to find out the count of positive numbers and negative numbers from a series of signed numbers in 8086.	66
22.	Write an assembly language program to convert to find out the largest number from a given unordered array of 8-bit numbers, stored in the locations starting from a known address in 8086.	67
23.	Write an assembly language program to convert to find out the largest number from a given unordered array of 16-bit numbers, stored in the locations starting from a known address in 8086.	68
24.	Write an assembly language program to print Fibonacci series in 8086.	69
25.	Write an assembly language program to perform the division 15/6 using the ASCII codes. Store the ASCII codes of the result in register DX.	70
	<b>Interfacing using 8086-Microprocessor Kit</b>	
26.	Interfacing ET-8255 study card with ET-8086 microprocessor hardware kit.	71
27.	Interfacing ET-8253 study card with ET-8086 microprocessor hardware kit.	73
28.	Interfacing ET-8279 study card with ET-8086 microprocessor hardware kit.	75
29.	Interfacing ET-8251 study card with ET-8086 microprocessor hardware kit.	77
30.	Interfacing ET-8259 study card with ET-8086 microprocessor hardware kit.	79

# QUESTION NO: 1

## OBJECTIVE:

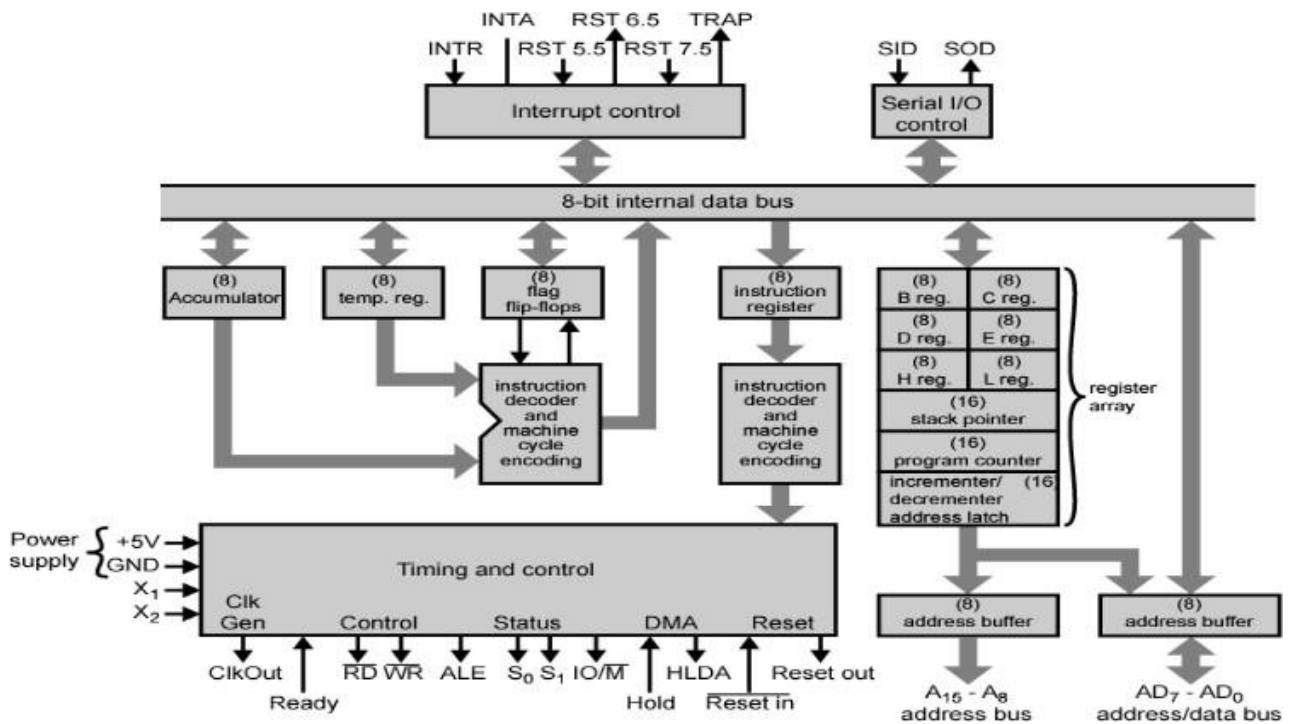
Introduction of 8085-microprocessor kit and steps for execution on the kit.

The Intel 8085 microprocessor is an 8-bit microprocessor introduced by Intel in 1976. It is part of the 8080-microprocessor family and is often considered an enhanced version of the Intel 8080. The 8085 microprocessor was widely used in early microcomputer systems and embedded applications. Key features of the Intel 8085 microprocessor include:

- **ARCHITECTURE:** 8-bit data bus, 16-bit address bus, addressing up to 64 KB memory, single accumulator for arithmetic/logic operations.
- **REGISTERS:** Accumulator (ACC), general-purpose (B, C, D, E, H, L), special-purpose (SP, PC).
- **CLOCK SPEED:** Operates at 3 MHz (varying in different versions).
- **INSTRUCTION SET:** Based on 8080, enhanced with additional instructions, supports data manipulation, control, and I/O operations.
- **ADDRESSING MODES:** Immediate, direct, indirect, and register addressing modes.
- **INTERRUPTS:** Supports TRAP, RST 7.5, RST 6.5, RST 5.5, INTR interrupt lines.
- **PERIPHERAL INTERFACE:** Interfaces with peripherals through I/O instructions and ports.
- **POWER SUPPLY:** Typically operates with a +5V power supply.



**Fig. 1.1:** Intel 8085 Microprocessor Training Kit



**Fig. 1.2:** Block Diagram of Intel 8085 Microprocessor

## 1. Address Bus (Pin 21-28)

- 16 bit address lines A0 to A15.
- The address bus has 8 signal lines A8 – A15 which are unidirectional.
- The other 8 address lines A0 to A7 are multiplexed (time shared) with the 8 data bits.

## 2. Data Bus (Pin 19-12)

- To save the number of pins lower order address pin are multiplexed with 8 bit data bus (bidirectional)
- So, the bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time.
- During the execution of the instruction, these lines carry the address bits during the early part (T1 state), then during the late parts (T2 state) of the execution, they carry the 8 data bits.

### 3. Status Pins – ALE, S1, S0

#### a) ALE(Address Latch Enable) : (Pin 30)

- i. Used to demultiplexed the address and data bus.
- ii. +ve going pulse generated when a new operation is started by microprocessor.
  - iii. ALE = 1 when the AD0 – AD7 lines have an address.
  - iv. ALE = 0 When it is low it indicates that the contents are data.
- v. This signal can be used to enable a latch to save the address bits from the AD lines.

#### b) S1 and S0 (Status Signal) : (Pin 33 and 29)

- i. Status signals to specify the kind of operation being performed .
- ii. Usually un-used in small systems.

S1	S0	Operations
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

**Table 1.1:** Status Signals

### 4. Control Pins – RD, WR, IO/M (active low)

#### a) RD: Read (Active low) (Pin 32)

- i. Read Memory or I/O device.
- ii. Indicated that data is to be read either from memory or I/P device and data bus is ready for accepting data from the memory or I/O device.

#### b) WR: Write (Active low) (Pin 31)

- i. Write Memory or I/O device
- ii. Indicated that data on the data bus are to be written into selected memory or I/P device.

#### c) IO/M: (Input Output/Memory-Active low) (Pin 34)

- i. Signal specifies that the read/write operation relates to whether memory or I/O device.
- ii. When (IO/M=1) the address on the address bus is for I/O device.
- iii. When (IO/M=0) the address on the address bus is for memory.

IO/M(active low)	RD	WR	Control Signals	Operations
0	0	1	MEMR	M/M Read
0	1	0	MEMW	M/M Write
1	0	1	IOR	I/O Read
1	1	0	IOW	I/O Write

**Table 1.2:** Control Signals

IO/M	S1	S0	Operations	Control Signals
0	1	1	Opcode Fetch	$RD=0$
0	1	0	Memory Read	$RD=0$
0	0	1	Memory Write	$WR=0$
1	1	0	I/O Read	$RD=0$
1	0	1	I/O Write	$WR=0$
1	1	0	Interrupt Acknowledge	$INTA=0$
Z	0	0	Halt	$RD, WR=Z$ & $INTA=1$
Z	X	X	Hold	
Z	X	X	Reset	

**Table 1.3:** Control & Status Signals

## 5. Interrupt Pins (Pin 6-11)

- They are the signals initiated by an external device to request the microprocessor to do a particular task or work.
- On receipt of an interrupt, the microprocessor acknowledges the interrupt by the active low INTA (Interrupt Acknowledge) signal.

## **6. Power Supply & Clock Signal**

### **a) Vcc : (Pin 40)**

- i. Single +5 volt power supply

### **b) Vss : (Pin 20)**

- i. Ground.

### **c) X0 and X1 : (Pin 1-2)**

- i. Crystal or R/C network or LC network connections to set the frequency of internal clock generator.
- ii. The frequency is internally divided by two.
- iii. Since the basic operating timing frequency is 3 MHz, a 6 MHz crystal is connected to the X0 and X1 pins.

### **d) CLK (output) : (Pin 37)**

- i. Clock Output is used as the system clock for peripheral and devices interfaced with the microprocessor.

## **7. HOLD (Pin 38)**

- a) This signal indicates that another device is requesting the use of address and data bus.
- b) So it relinquish the use of buses as soon as the current machine cycle is completed.
- c) Microprocessor regains the bus after the removal of a HOLD signal.

## **8. HLDA (Pin 39)**

- a) On receipt of HOLD signal, the MP acknowledges the request by sending out HLDA signal and leaves out the control of the buses.
- b) After the HLDA signal the DMA controller starts the direct transfer of data.
- c) After the removal of HOLD request HLDA goes low.

## **9. SID (input) Serial input data (Pin 4)**

- a) It is a data line for serial input.
- b) Used to accept serial data bit by bit from external device.
- c) The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

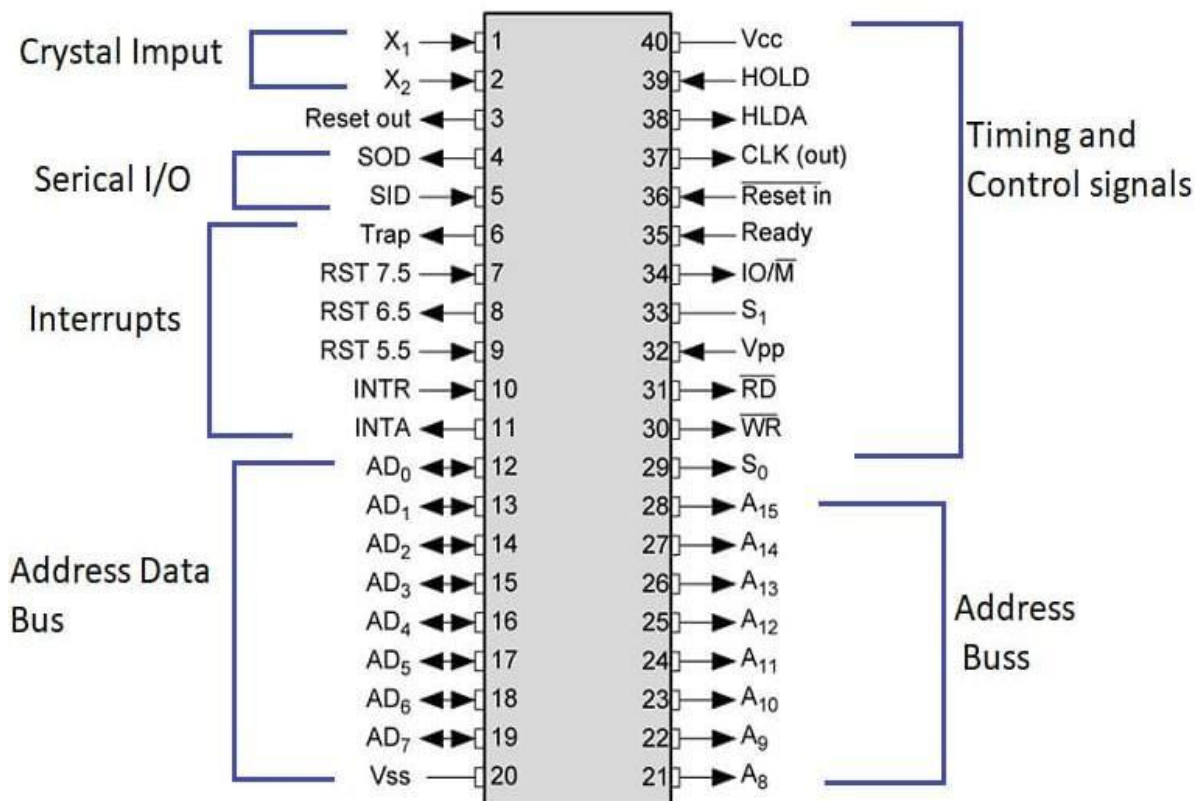
## **10. SOD (output) Serial output data (Pin 5)**

- a) It is a data line for serial output.
- b) Used to transmit serial data bit by bit to the external device.
- c) The 7th bit of the accumulator is outputted on SOD line when SIM instruction is executed.



## 11. Ready (input) (Pin 35)

- a) Memory and I/O devices will have slower response compared to microprocessors.
- b) Before completing the present job such a slow peripheral may not be able to handle further data or control signals from CPU.
- c) The processor sets the READY signal after completing the present job to access the data.
- d) It synchronize slower peripheral to the processor.

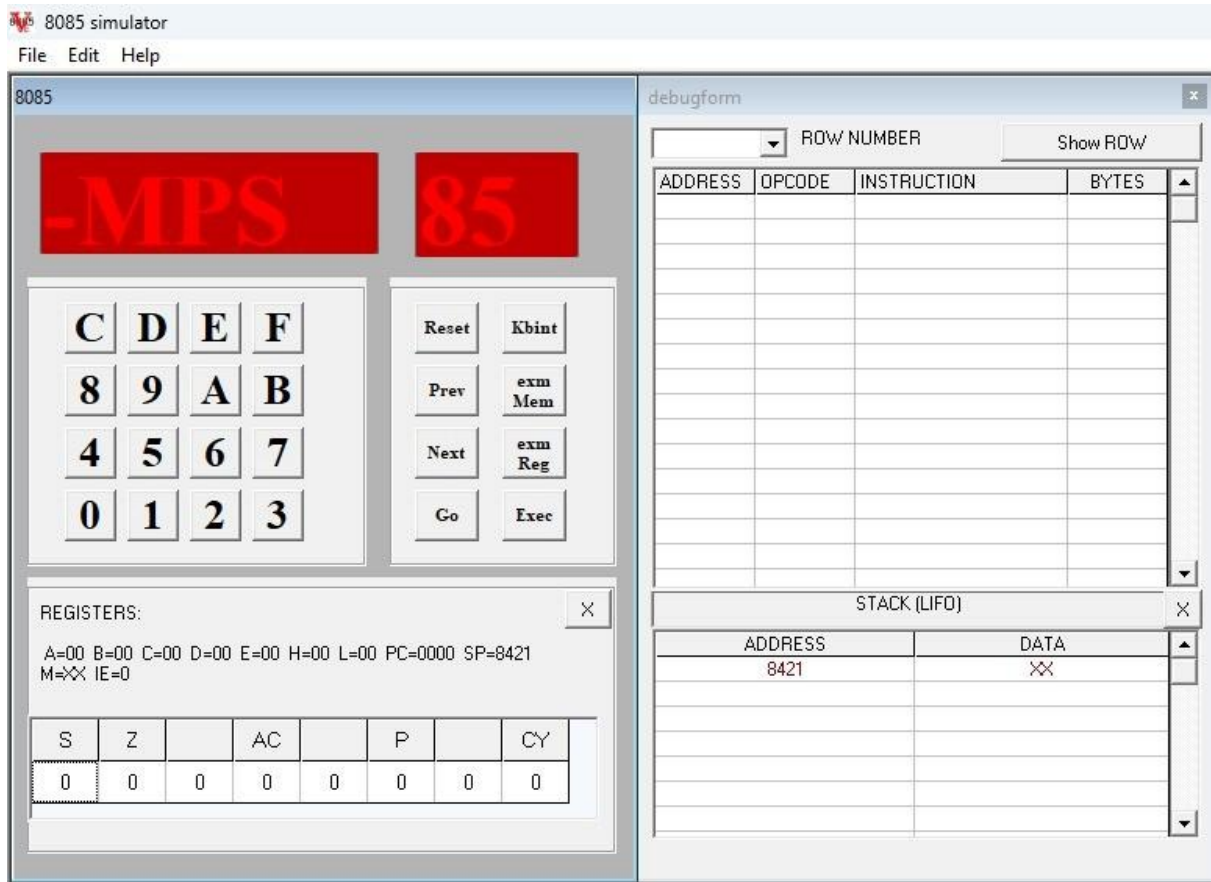


**Fig. 1.3:** Pin Diagram of Intel 8085 Microprocessor

## **FEATURES of MICROPROCESSOR – 8085**

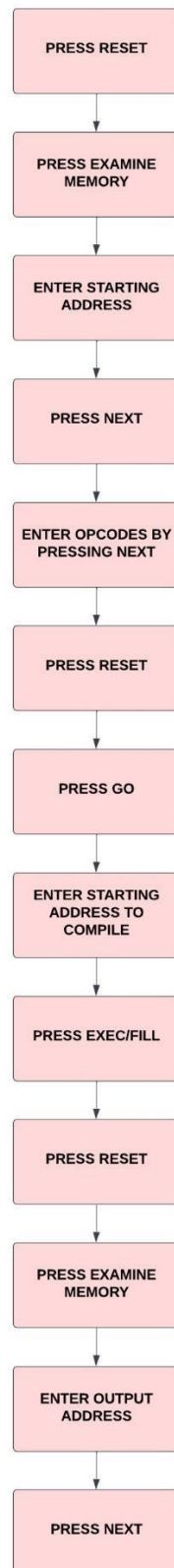
1. 8085 is developed by INTEL.
2. 8-bit microprocessor: can accept 8-bit data simultaneously.
3. Operates on single +5V D.C. supply.
4. Designed using NMOS technology.
5. 6200 transistors on a single chip.
6. It provides on chip clock generator; hence it does not require external clock generator.
7. Operates on 3MHz clock frequency.
8. 8bit multiplexed address/data bus, which reduce the number of pins.
9. 16address lines, hence it can address  $2^{16} = 64$  K bytes of memory
10. It generates 8 bit I/O addresses; hence it can access  $2^8 = 256$  I/O ports. 5 hardware interrupts i.e. TRAP/RST4.5, RST 7.5, RST 6.5, RST 5.5, and INTR
11. It provides DMA (Direct memory access).
12. 40-pin L.C. package fabricated on a single LSI chip.
13. Clock cycle is 320ns.
14. 80 basic instructions and 246 opcodes.

# VIKAS SIMULATOR



**Fig. 1.4:** Vikas Simulator

## **STEPS TO PERFORM ON VIKAS SIMULATOR**



**Fig. 1.5:** Steps to perform on Vikas Simulator

## **QUESTION NO: 2(i)**

### **OBJECTIVE:**

Write a program to store 8-bit data into one register and then copy that to all registers.

CODE	MEMORY LOCATION	OPCODE
MVI A, 48	8000, 8001	3E, 48
MOV B, A	8002	47
MOV C, A	8003	4F
MOV D, A	8004	57
MOV E, A	8005	5F
MOV H, A	8006	67
MOV L, A	8007	6F
RST 5	8008	EF

**Table 2(i).1:** Code Explanation

REGISTERS: <span>×</span>							
A=48 B=48 C=48 D=48 E=48 H=48 L=48 PC=8009 SP=8421 M=XX IE=0							
S	Z		AC		P		CY
0	0	0	0	0	0	0	0

**Fig. 2(i).1:** Register Output



## **QUESTION NO: 2(ii)**

### **OBJECTIVE:**

Write a program for addition of two 8-bit numbers.

CODE	MEMORY LOCATION	OPCODE
MVI A, 48	8000, 8001	3E, 48
MVI B, 48	8002, 8003	06, 48
ADD B	8004	80
STA 8500	8005, 8006, 8007	32, 00, 85
RST 5	8008	EF

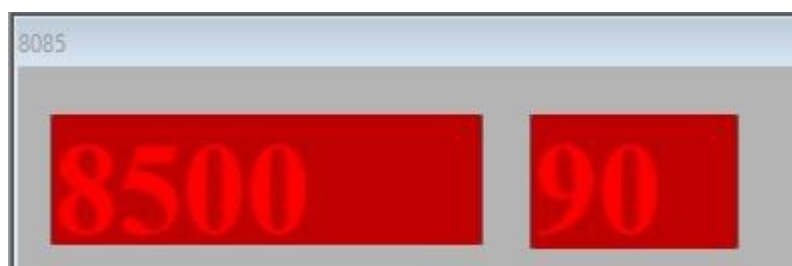
**Table 2(ii).1:** Code Explanation

REGISTERS:							
A=90 B=48 C=00 D=00 E=00 H=00 L=00 PC=8009 SP=8421 M=XX IE=0							
S	Z		AC		P		CY
1	0	0	1	0	1	0	0

**Fig. 2(ii).1:** Register Output

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	3E	MVI A,8 bit	2
8001	48		
8002	06	MVI B,8 bit	2
8003	48		
8004	80	ADD B	1
8005	32	STA 16 bit	3
8006	00		
8007	85		
8008	EF	RST 5	1

**Fig. 2(ii).2:** Debugform



**Fig. 2(ii).3:** Output

**OUTPUT –**

[ 8500] – 90



## **QUESTION NO: 2(iii)**

### **OBJECTIVE:**

Write a program to add 8-bit numbers using direct and indirect addressing mode.

#### **A. Direct Addressing**

<b>CODE</b>	<b>MEMORY LOCATION</b>	<b>OPCODE</b>
LDA 8500	8000, 8001, 8002	3A, 00, 85
MOV B, A	8003	47
LDA 8501	8004, 8005, 8006	3A, 01, 85
ADD B	8007	80
STA 8502	8008, 8009, 800A	32, 02, 85
RST 5	800B	EF

**Table 2(iii).1: Code Explanation**

#### **B. In-Direct Addressing**

<b>CODE</b>	<b>MEMORY LOCATION</b>	<b>OPCODE</b>
LXI H, 8500	8000, 8001, 8002	21, 00, 85
MOV A, M	8003	7E
INX H	8004	23
ADD M	8005	86
INX H	8006	23
MOV M, A	8007	77
RST 5	8008	EF

**Table 2(iii).2: Code Explanation**

REGISTERS:							
A=C0 B=00 C=00 D=00 E=00 H=85 L=02 PC=8009 SP=8421 M=C0 IE=0							
S	Z		AC		P		CY
1	0	0	1	0	1	0	0

**Fig. 2(iii).1:** Register Output

<div> <div></div> <div>ROW NUMBER</div> <div>Show ROW</div> </div>			
ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	21	LXI H,16 bit	3
8001	00		
8002	85		
8003	7E	MOV A,M	1
8004	23	INX H	1
8005	86	ADD M	1
8006	23	INX H	1
8007	77	MOV M,A	1
8008	EF	RST 5	1

**Fig. 2(iii).2:** Debugform



**Fig. 2(iii).3: Output**

**INPUT –**

[ 8500] – 58, [ 8501] – 68

**OUTPUT –**

[ 8502] – C0

## **QUESTION NO: 2(iv)**

### **OBJECTIVE:**

Write a program to add 16-bit numbers using direct and indirect addressing mode.

#### **A. Direct Addressing**

CODE	MEMORY LOCATION	OPCODE
LHLD 8500	8000, 8001, 8002	2A, 00, 85
XCHG	8003	EB
LHLD 8502	8004, 8005, 8006	2A, 02, 85
DAD D	8007	19
SHLD 8504	8008, 8009, 800A	22, 04, 85
RST 5	800B	EF

**Table 2(iv).1: Code Explanation**

REGISTERS:							
A=00 B=00 C=00 D=48 E=48 H=90 L=90 PC=800C SP=8421 M=00 IE=0							
S	Z		AC		P		CY
0	0	0	0	0	0	0	0

**Fig 2(iv).1: Register Output**

debugform

ROW NUMBER Show ROW

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	2A	LHLD 16 bit	3
8001	00		
8002	85		
8003	EB	XCHG	1
8004	2A	LHLD 16 bit	3
8005	02		
8006	85		
8007	19	DAD D	1
8008	22	SHLD 16 bit	3
8009	04		
800A	85		
800B	EF	RST 5	1

**Fig 2(iv).2:** Debugform

8085	8504	90	8085	8505	90
------	------	----	------	------	----

**Fig 2(iv).3:** Output

### INPUT –

[ 8500] – 48, [ 8501] – 48, [ 8502] – 48, [ 8503] – 48

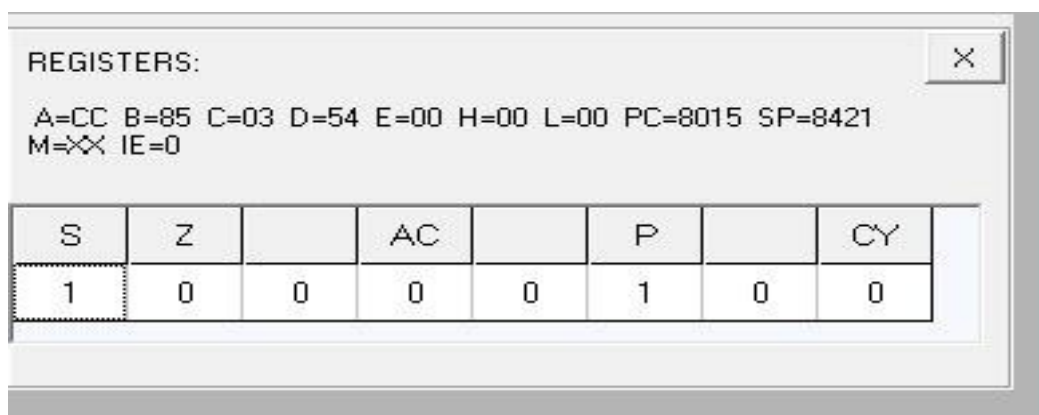
### OUTPUT –

[ 8504] – 90, [ 8505] – 90

## B. In-Direct Addressing

CODE	MEMORY LOCATION	OPCODE
LXI B, 8500	8000, 8001, 8002	01, 00, 85
LDAX B	8003	0A
MOV D, A	8004	57
INX B	8005	03
LDAX B	8006	0A
ADD D	8007	82
STA 8504	8008, 8009, 800A	32, 04, 85
INX B	800B	03
LDAX B	800C	0A
MOV D, A	800D	57
INX B	800E	03
LDAX B	800F	0A
ADC D	8010	8A
STA 8505	8011, 8012, 8013	32, 05, 85
RST 5	8014	EF

**Table 2(iv).2:** Code Explanation



REGISTERS:

A=CC B=85 C=03 D=54 E=00 H=00 L=00 PC=8015 SP=8421  
M=XX IE=0

S	Z		AC		P		CY
1	0	0	0	0	1	0	0

**Fig. 2(iv).4:** Register Output

debugform			
		ROW NUMBER	Show ROW
ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	01	LXI B,16 bit	3
8001	00		
8002	85		
8003	0A	LDAX B	1
8004	57	MOV D,A	1
8005	03	INX B	1
8006	0A	LDAX B	1
8007	82	ADD D	1
8008	32	STA 16 bit	3
8009	04		
800A	85		
800B	03	INX B	1
800C	0A	LDAX B	1
800D	57	MOV D,A	1
800E	03	INX B	1
800F	0A	LDAX B	1
8010	8A	ADC D	1
8011	32	STA 16 bit	3
8012	05		
8013	85		
8014	EF	RST 5	1

**Fig. 2(iv).5:** Debugform

8085	8504	7C
8085	8505	CC

**Fig. 2(iv).6:** Output

### INPUT –

[ 8500] – 34, [ 8501] – 48, [ 8502] – 54, [ 8503] – 78

### OUTPUT –

[ 8504] – 7C [ 8505] – CC

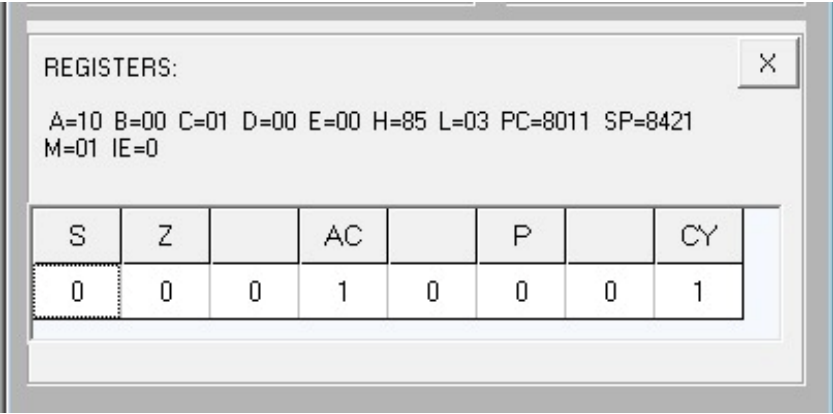
## **QUESTION NO: 2(v)**

### **OBJECTIVE:**

Write a program to 8-bit numbers using carry. (using JNC instruction).

CODE	MEMORY LOCATION	OPCODE
MVIC, 00	8000,8001	0E,00
LXI H, 8500	8002,8003,8004	21,00,85
MOV A, M	8005	7E
INX H	8006	23
ADD M	8007	86
JNC Next	8008,8009,800A	D2,0C,80
INR C	800B	0C
Next: INX H	800C	23
MOV M, A	800D	77
INX H	800E	23
MOV M, C	800F	71
RST 5	8010	EF

**Table 2(v).1:** Code Explanation



REGISTERS:							
A=10 B=00 C=01 D=00 E=00 H=85 L=03 PC=8011 SP=8421 M=01 IE=0							
S	Z		AC		P		CY
0	0	0	1	0	0	0	1

**Fig. 2(v).1:** Register Output



debugform

ROW NUMBER Show ROW

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	0E	MVI C,8 bit	2
8001	00		
8002	21	LXI H,16 bit	3
8003	00		
8004	85		
8005	7E	MOV A,M	1
8006	23	INX H	1
8007	86	ADD M	1
8008	D2	JNC 16 bit	3
8009	0C		
800A	80		
800B	0C	INR C	1
800C	23	INX H	1
800D	77	MOV M,A	1
800E	23	INX H	1
800F	71	MOV M,C	1
8010	EF	RST 5	1

**Fig. 2(v).2:** Debugform



**Fig. 2(v).3:** Output

**INPUT –**

[ 8500] – 88, [ 8501] – 88

**OUTPUT –**

[ 8502] – 10, [ 8503] – 01

## **QUESTION NO: 2(vi)**

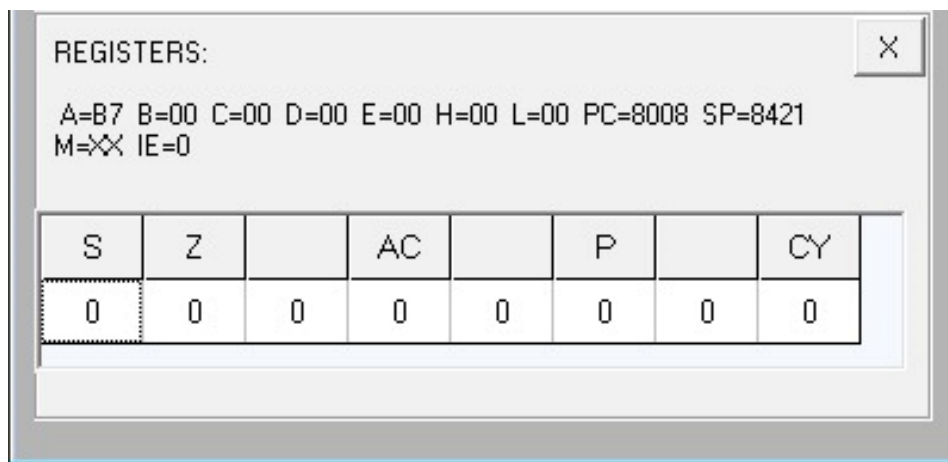
### **OBJECTIVE:**

Write a program to find 1's complement and 2's complement of 8-bit number.

#### **A. 1's Compliment**

CODE	MEMORY LOCATION	OPCODE
LDA 8500H	8000, 8001, 8002	3A, 00, 85
CMA	8003	2F
STA 8501H	8004,8005,8006	32, 01, 85
RST 5	8007	EF

**Table 2(vi).1: Code Explanation**



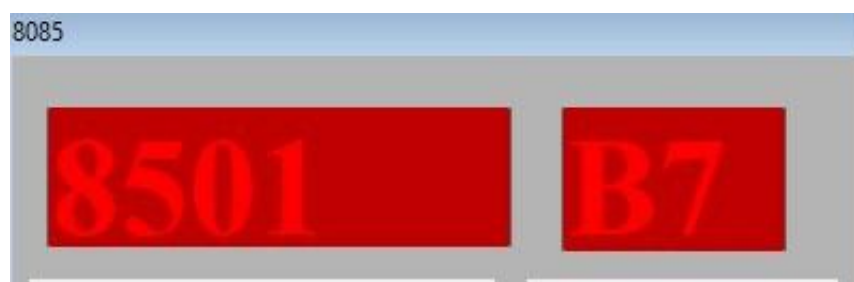
**Fig 2(vi).1: Register Output**

debugform

ROW NUMBER Show ROW

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	3A	LDA 16 bit	3
8001	00		
8002	85		
8003	2F	CMA	
8004	32	STA 16 bit	3
8005	01		
8006	85		
8007	EF	RST 5	1

**Fig 2(vi).2:** Debugform



**Fig 2(vi).3:** Output

**INPUT –**

[ 8500] – 48

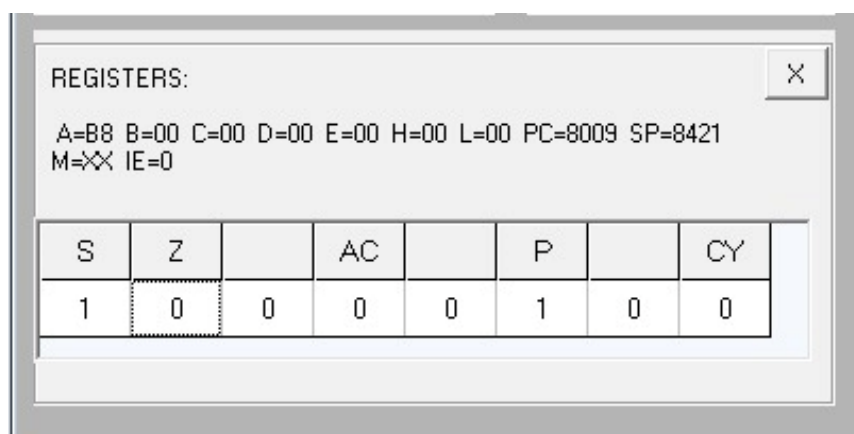
**OUTPUT –**

[ 8501] – B7

## B. 2's Compliment

CODE	MEMORY LOCATION	OPCODE
LDA 8500H	8000, 8001, 8002	3A, 00, 85
CMA	8003	2F
INR A	8004	3C
STA 8501H	8005,8006,8007	32, 01, 85
RST 5	8008	EF

**Table 2(vi).2:** Code Explanation



REGISTERS:

A=B8 B=00 C=00 D=00 E=00 H=00 L=00 PC=8009 SP=8421  
M=XX IE=0

S	Z		AC		P		CY
1	0	0	0	0	1	0	0

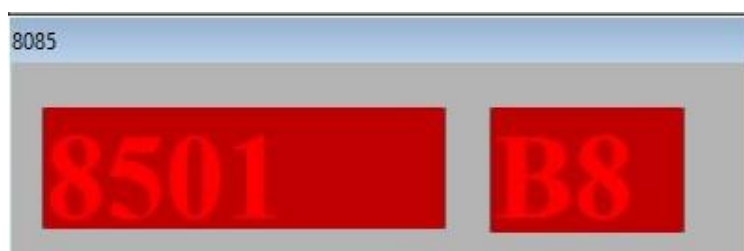
**Fig. 2(vi).4:** Register Output

debugform

ROW NUMBER Show ROW

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	3A	LDA 16 bit	3
8001	00		
8002	85		
8003	2F	CMA	
8004	3C	INR A	1
8005	32	STA 16 bit	3
8006	01		
8007	85		
8008	EF	RST 5	1

**Fig. 2(vi).5:** Debugform



**Fig. 2(vi).6:** Output

**INPUT –**

[ 8500] – 48

**OUTPUT –**

[ 8501] – B8

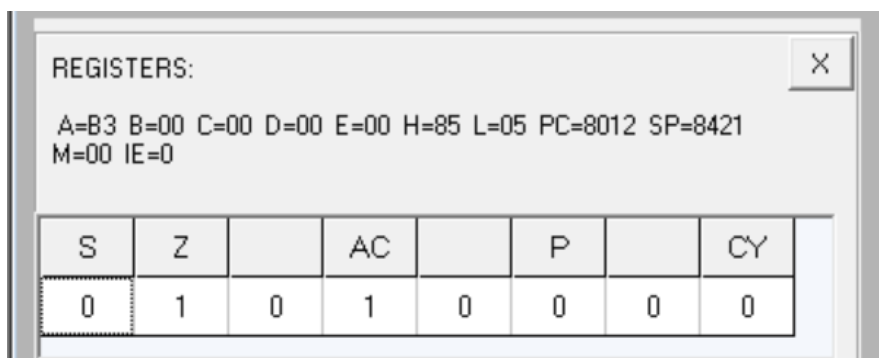
## QUESTION NO: 3

### OBJECTIVE:

Write a program for the sum of series of numbers.

CODE	MEMORY LOCATION	OPCODE
LDA 8500H	8000,8001,8002	3A,00,85
MOV C, A	8003	4F
SUB A	8004	97
LXI H, 8501H	8005,8006,8007	21,01,85
Back: ADD M	8008	86
INX H	8009	23
DCR C	800A	0D
JNZ Back	800B,800C,800D	C2,08,80
STA 8600H	800E,800F,8010	32,00,86
RST 5	8011	EF

**Table 3.1:** Code Explanation



REGISTERS:

A=B3 B=00 C=00 D=00 E=00 H=85 L=05 PC=8012 SP=8421  
M=00 IE=0

S	Z		AC		P		CY
0	1	0	1	0	0	0	0

**Fig. 3.1:** Register Output

debugform

ROW NUMBER Show ROW

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	3A	LDA 16 bit	3
8001	00		
8002	85		
8003	4F	MOV C,A	1
8004	97	SUB A	1
8005	21	LXI H,16 bit	3
8006	01		
8007	85		
8008	86	ADD M	1
8009	23	INX H	1
800A	0D	DCR C	1
800B	C2	JNZ 16 bit	3
800C	08		
800D	80		
		JUMPED TO 8008	

Fig. 3.2: Debugform



Fig. 3.3: Output

### INPUT –

[8500] – 04, [8501] – 9A, [8502] – 52, [8503] – 89, [8504] – 3E

### RESULT –

1B3

### OUTPUT –

B3

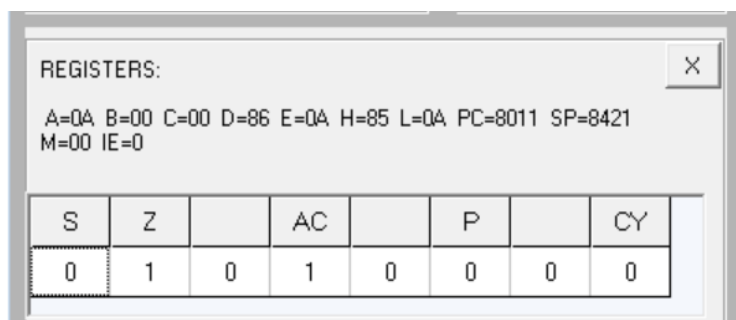
## QUESTION NO: 4

### OBJECTIVE:

Write a program for data transfer from memory block B1 to memory block B2.

CODE	MEMORY LOCATION	OPCODE
MVI C, 0AH	8000,8001	0E,0A
LXI H, 8500H	8002,8003,8004	21,00,85
LXI D, 8600H	8005,8006,8007	11,00,86
Back: MOV A, M	8008	7E
STAX D	8009	12
INX H	800A	23
INX D	800B	13
DCR C	800C	0D
JNZ Back	800D,800E,800F	C2,08,80
RST 5	8010	EF

**Table 4.1:** Code Explanation



REGISTERS:

A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421  
M=00 IE=0

S	Z		AC		P		CY
0	1	0	1	0	0	0	0

**Fig. 4.1:** Register Output



debugform			
50	ROW NUMBER	Show ROW	
ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	0E	MVI C,8 bit	2
8001	0A		
8002	21	LXI H,16 bit	3
8003	00		
8004	85		
8005	11	LXI D,16 bit	3
8006	00		
8007	86		
8008	7E	MOV A,M	1
8009	12	STAX D	1
800A	23	INX H	1
800B	13	INX D	1
800C	0D	DCR C	1
800D	C2	JNZ 16 bit	3
800E	08		
800F	80		
STACK (LIFO)			

Fig. 4.2: Debugform



Fig. 4.3: Output

## **INPUT –**

[8500] – 01, [8501] – 02, [8502] – 03..... [8509] – 0A

## **OUTPUT –**

[8600] – 01, [8601] – 02, [8602] – 03...      [8609] – 0A

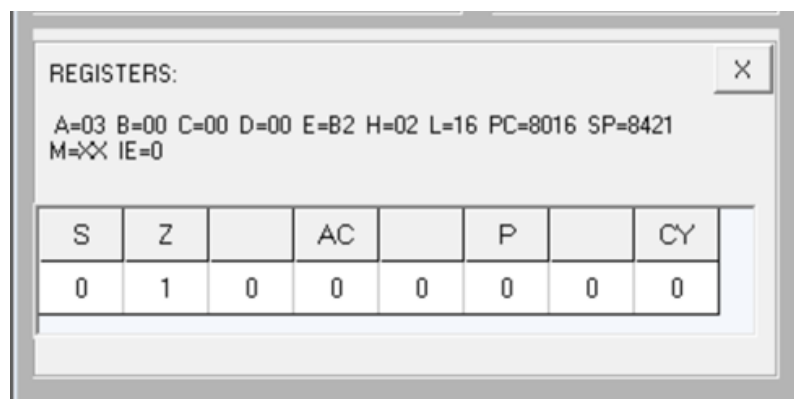
## QUESTION NO: 5

### OBJECTIVE:

Write a program for multiply two 8-bit numbers.

CODE	MEMORY LOCATION	OPCODE
LDA 8500H	8000,8001,8002	3A,00,85
MOV E, A	8003	5F
MVI D, 00	8004,8005	16,00
LDA 8501H	8006,8007,8008	3A,01,85
MOV C, A	8009	4F
LXI H, 0000H	800A,800B,800C	21,00,00
Back: DAD D	800D	19
DCR C	800E	0D
JNZ Back	800F,8010,8011	C2,0D,80
SHLD 8600H	8012,8013,8014	22,00,86
RST 5	8015	EF

**Table 5.1:** Code Explanation



REGISTERS:							
A=03 B=00 C=00 D=00 E=B2 H=02 L=16 PC=8016 SP=8421							
M=XX IE=0							
S	Z		AC		P		CY
0	1	0	0	0	0	0	0

**Fig. 5.1:** Register Output

debugform

ROW NUMBER Show ROW

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	3A	LDA 16 bit	3
8001	00		
8002	85		
8003	5F	MOV E,A	1
8004	16	MVI D,8 bit	2
8005	00		
8006	3A	LDA 16 bit	3
8007	01		
8008	85		
8009	4F	MOV C,A	1
800A	21	LXI H,16 bit	3
800B	00		
800C	00		
800D	19	DAD D	1
800E	0D	DCR C	1
800F	C2	JNZ 16 bit	3

STACK (LIFO)

**Fig. 5.2:** Debugform

8085	8600	16
8085	8601	02

**Fig. 5.3:** Output

### INPUT –

[8500] – B2, [8501] – 03

### RESULT–

$B2 + B2 + B2 = 0216\text{ H}$

### OUTPUT –

[8600] – 16, [8601] – 02

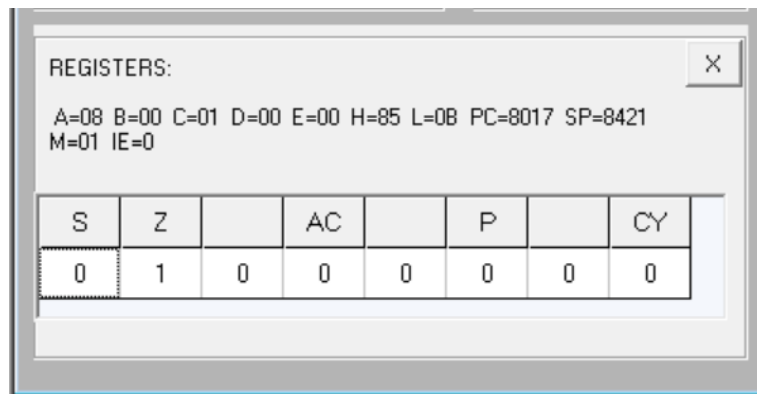
## **QUESTION NO: 6**

### **OBJECTIVE:**

Write a program to add ten 8-bit numbers. Assume the numbers are stored in 8500-8509. Store the result in 850A and 850B memory address.

<b>CODE</b>	<b>MEMORY LOCATION</b>	<b>OPCODE</b>
<b>MVI C, 00</b>	8000,8001	0E, 00
<b>MVI B, 09</b>	8002,8003	06, 09
<b>LXI H, 8500H</b>	8004,8005,8006	21,00,85
<b>MOV A, M</b>	8007	7E
<b>Back: INX H</b>	8008	23
<b>ADD M</b>	8009	86
<b>JNC Next</b>	800A,800B,800C	D2, 0E,80
<b>INR C</b>	800D	0C
<b>Next: DCR B</b>	800E	05
<b>JNZ Back</b>	800F,8010,8011	C2,08,80
<b>INX H</b>	8012	23
<b>MOV M, A</b>	8013	77
<b>INX H</b>	8014	23
<b>MOV M, C</b>	8015	71
<b>RST 5</b>	8016	EF

**Table 6.1:** Code Explanation



**Fig. 6.1:** Register Output

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	0E	MVI C,8 bit	2
8001	00		
8002	06	MVI B,8 bit	2
8003	09		
8004	21	LXI H,16 bit	3
8005	00		
8006	85		
8007	7E	MOV A,M	1
8008	23	INX H	1
8009	86	ADD M	1
800A	D2	JNC 16 bit	3
800B	0E		
800C	80		
800D	0C	INR C	1
800E	05	DCR B	1
800F	C2	JNZ 16 bit	3

**Fig. 6.2:** Debugform



**Fig. 6.3:** Output

## INPUT –

[8500] – FF, [8501] – 01, [8502] – 01, [8503] – 01, [8504] – 01, [8505] – 01, [8506] – 01, [8507] – 01, [8508] – 01, [8509] – 01

## OUTPUT –

[850A] – 08, [850B] – 01

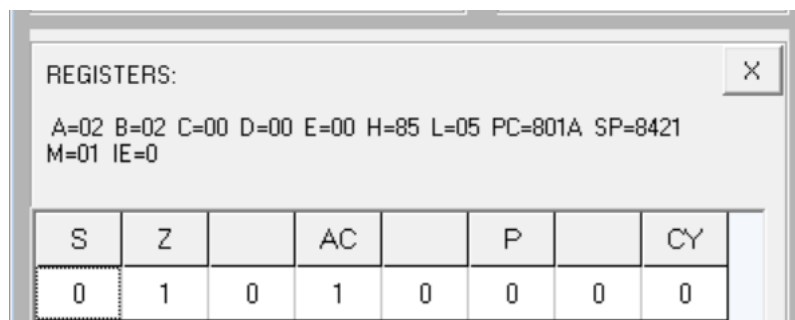
## QUESTION NO: 7

### OBJECTIVE:

Write a program to find the negative numbers in a block of data.

CODE	MEMORY LOCATION	OPCODE
LDA 8500H	8000,8001,8002	3A,00,85
MOV C, A	8003	4F
MVI B, 00	8004,8005	06, 00
LXI H, 8501H	8006,8007,8008	21,01,85
Back: MOV A, M	8009	7E
ANI 80H	800A,800B	E6,80
JZ Skip	800C,800D,800E	CA,10,80
INR B	800F	04
Skip: INX H	8010	23
DCR C	8011	0D
JNZ Back	8012,8013,8014	C2,09,80
MOV A, B	8015	78
STA 8600H	8016,8017,8018	32,00,86
RST 5	8019	EF

**Table 7.1:** Code Explanation



REGISTERS:

A=02 B=02 C=00 D=00 E=00 H=85 L=05 PC=801A SP=8421  
M=01 IE=0

S	Z		AC		P		CY
0	1	0	1	0	0	0	0

**Fig. 7.1:** Register Output

debugform

ROW NUMBER Show ROW

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	3A	LDA 16 bit	3
8001	00		
8002	85		
8003	4F	MOV C,A	1
8004	06	MVI B,8 bit	2
8005	00		
8006	21	LXI H,16 bit	3
8007	01		
8008	85		
8009	7E	MOV A,M	1
800A	E6	ANI 8 bit	2
800B	80		
800C	CA	JZ 16 bit	3
800D	10		
800E	80		
		JUMPED TO 8010	
STACK (LIFO)			

**Fig. 7.2:** Debugform



**Fig. 7.3:** Output

## INPUT –

[8500] – 04, [8501] – 56, [8502] – A9, [8503] – 73, [8504] – 82

## OUTPUT –

02



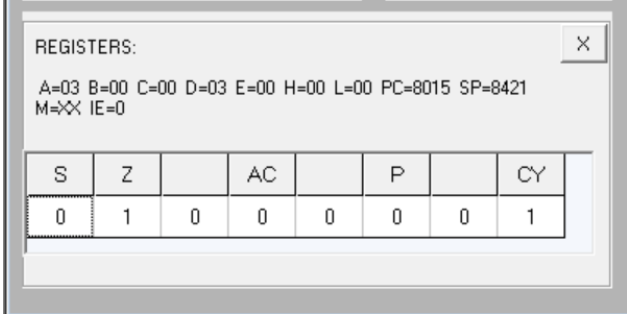
## QUESTION NO: 8

### OBJECTIVE:

Write a program to count the number of one's in a number.

CODE	MEMORY LOCATION	OPCODE
LDA 8500H	8000,8001,8002	3A,00,85
MVI B, 08	8003,8004	06,08
MVI D, 00	8005,8006	16,00
Loop1: RLC	8007	07
JNC Loop2	8008,8009,800A	D2,0C,80
INR D	800B	14
Loop2: DCR B	800C	05
JNZ Loop1	800D,800E,800F	C2,07,80
MOV A, D	8010	7A
STA 8600H	8011,8012,8013	32,00,86
RST 5	EF	EF

**Table 8.1:** Code Explanation



REGISTERS:

A=03 B=00 C=00 D=03 E=00 H=00 L=00 PC=8015 SP=8421  
M=XX IE=0

S	Z		AC		P		CY
0	1	0	0	0	0	0	1

**Fig. 8.1:** Register Output

debugform

ROW NUMBER Show ROW

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	3A	LDA 16 bit	3
8001	00		
8002	85		
8003	06	MVI B,8 bit	2
8004	08		
8005	16	MVI D,8 bit	2
8006	00		
8007	07	RLC	1
8008	D2	JNC 16 bit	3
8009	0C		
800A	80		
		JUMPED TO 800C	
800C	05	DCR B	1
800D	C2	JNZ 16 bit	3
800F	07		

STACK (LIFO)

**Fig. 8.2:** Debugform



**Fig. 8.3:** Output

**INPUT –**

[8500] – 25 0010 0101

**OUTPUT –**

[8600] – 03

## **QUESTION NO: 9**

### **OBJECTIVE:**

Write a program to arrange numbers in Ascending order.

<b>CODE</b>	<b>MEMORY LOCATION</b>	<b>OPCODE</b>
<b>LXI H, 8500H</b>	8000,8001,8002	21,00,85
<b>MOV C, M</b>	8003	4E
<b>DCR C</b>	8004	0D
<b>Repeat: MOV D, C</b>	8005	51
<b>LXI H, 8501H</b>	8006,8007,8008	21,01,85
<b>Loop: MOV A, M</b>	8009	7E
<b>INX H</b>	800A	23
<b>CMP M</b>	800B	BE
<b>JC Skip</b>	800C,800D,800E	DA,14,80
<b>MOV B, M</b>	800F	46
<b>MOV M, A</b>	8010	77
<b>DCX H</b>	8011	2B
<b>MOV M, B</b>	8012	70
<b>INX H</b>	8013	23
<b>Skip: DCR D</b>	8014	15
<b>JNZ Loop</b>	8015,8016,8017	C2,09,80
<b>DCR C</b>	8018	0D
<b>JNZ Repeat</b>	8019,801A,801B	C2,05,80
<b>RST5</b>	<b>801C</b>	<b>EF</b>

**Table 8.1:** Code Explanation

REGISTERS:							
A=02 B=01 C=00 D=00 E=00 H=85 L=02 PC=801D SP=8421 M=02 IE=0							
S	Z		AC		P		CY
0	1	0	0	0	0	0	0

**Fig. 8.1:** Register Output

debugform			
	ROW NUMBER	Show ROW	
ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	21	LXI H,16 bit	3
8001	00		
8002	85		
8003	4E	MOV C,M	1
8004	0D	DCR C	1
8005	51	MOV D,C	1
8006	21	LXI H,16 bit	3
8007	01		
8008	85		
8009	7E	MOV A,M	1
800A	23	INX H	1
800B	BE	CMP M	1
800C	DA	JC 16 bit	3
800D	14		
800E	80		
800F	46	MOV R,M	1
STACK (LIFO)			

**Fig. 8.2:** Debugform

8085	8500	05
8085	8501	01
8085	8502	02
8085	8503	03
8085	8504	04
8085	8505	05

**Fig. 8.3:** Output

## **INPUT –**

[8500] – 05, [8501] – 05, [8502] – 04, [8503] – 03, [8504] – 02, [8505] – 01

## **OUTPUT –**

[8500] – 05, [8501] – 01, [8502] – 02, [8503] – 03, [8504] – 04, [8505] – 05

## QUESTION NO: 10

### OBJECTIVE:

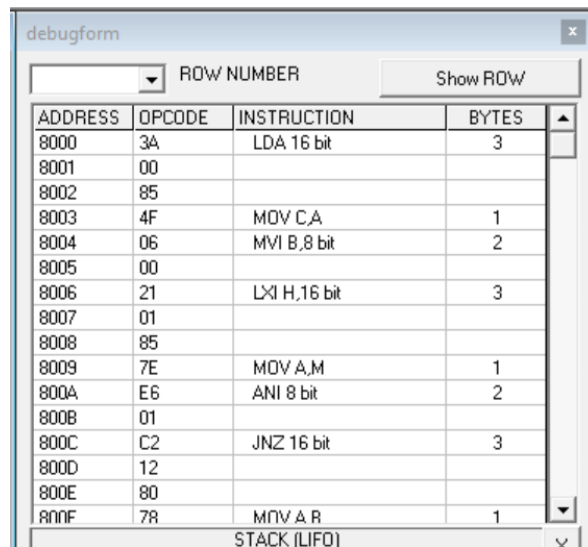
Write a program to calculate the sum of series of even numbers.

CODE	MEMORY LOCATION	OPCODE
LDA 8500H	8000,8001,8002	3A,00,85
MOV C, A	8003	4F
MVI B, 00	8004,8005	06 ,00
LXI H, 8501H	8006,8007,8008	21,01,85
Back: MOV A, M	8009	7E
ANI 01	800A,800B	E6,01
JNZ Skip	800C,800D,800E	C2,12,80
MOV A, B	800F	78
ADD M	8010	86
MOV B, A	8011	47
Skip: INX H	8012	23
DCR C	8013	0D
JNZ Back	8014,8015,8016	C2,09,80
STA 8600H	8017,8018,8019	32,00,86
RST 5	801A	EF

**Table 10.1:** Code Explanation

REGISTERS:							
A=42 B=42 C=00 D=00 E=00 H=85 L=05 PC=8018 SP=8421 M=05 IE=0							
S	Z		AC		P		CY
0	1	0	0	0	0	0	0

**Fig. 10.1:** Register Output



debugform

ROW NUMBER Show ROW

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	3A	LDA 16 bit	3
8001	00		
8002	85		
8003	4F	MOV C,A	1
8004	06	MVI B,8 bit	2
8005	00		
8006	21	LXI H,16 bit	3
8007	01		
8008	85		
8009	7E	MOV A,M	1
800A	E6	ANI 8 bit	2
800B	01		
800C	C2	JNZ 16 bit	3
800D	12		
800E	80		
800F	78	MOV A,R	1

STACK (LIFO)

**Fig. 10.2:** Debugform



**Fig. 10.3:** Output

## INPUT –

[8500] – 04, [8501] – 20, [8502] – 15 , [8503] – 13, [8504]–22

## OUTPUT –

[8600] – 42

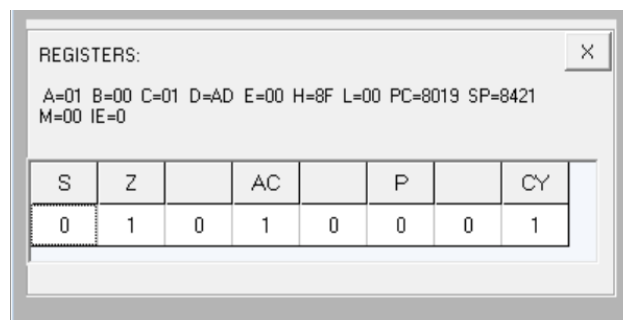
## QUESTION NO: 11

### OBJECTIVE:

Write an assembly language program to verify how many bytes are present in a given set, which resembles 10101101 in 8085.

CODE	MEMORY LOCATION	OPCODE
MVI B, 0A	8000,8001	06,0A
MVI D, AD	8002,8003	16, AD
MVI C, 00	8004,8005	0E,00
LXI H, 8500H	8006,8007,8008	21,00,85
Back: MOV A, M	8009	7E
CMP D	800A	BA
JNZ Next	800B,800C,800D	C2,0F,80
INR C	800E	0C
Next: INX H	800F	24
DCR B	8010	05
JNZ Back	8011,8012,8013	C2,09,80
MOV A, C	8014	79
STA 8600H	8015,8016,8017	32, 00,86
RST 5	8018	EF

**Table 11.1:** Code Explanation



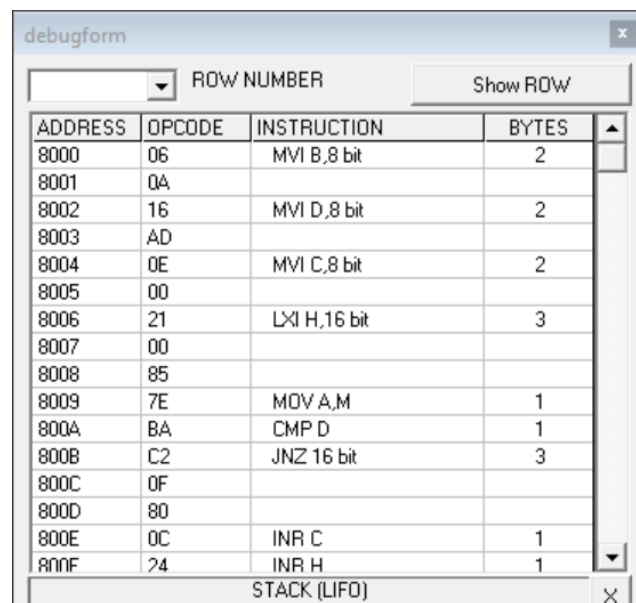
REGISTERS:

A=01 B=00 C=01 D=AD E=00 H=8F L=00 PC=8019 SP=8421  
M=00 IE=0

S	Z		AC		P		CY
0	1	0	1	0	0	0	1

**Fig. 11.1:** Register Output





debugform

ROW NUMBER Show ROW

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	06	MVI B,8 bit	2
8001	0A		
8002	16	MVI D,8 bit	2
8003	AD		
8004	0E	MVI C,8 bit	2
8005	00		
8006	21	LXI H,16 bit	3
8007	00		
8008	85		
8009	7E	MOV A,M	1
800A	BA	CMP D	1
800B	C2	JNZ 16 bit	3
800C	0F		
800D	80		
800E	0C	INR C	1
800F	24	INR H	1

STACK (LIFO)

**Fig. 11.2:** Debugform



**Fig. 11.3:** Output

## INPUT –

[8500] – AD, [8501] – 01, [8502] – 01, [8503] – 01, [8504] – 01, [8505] – 01, [8506] – 01, [8507] – 01, [8508] – 01, [8509] – 01

## OUTPUT –

[8600] – 01

---

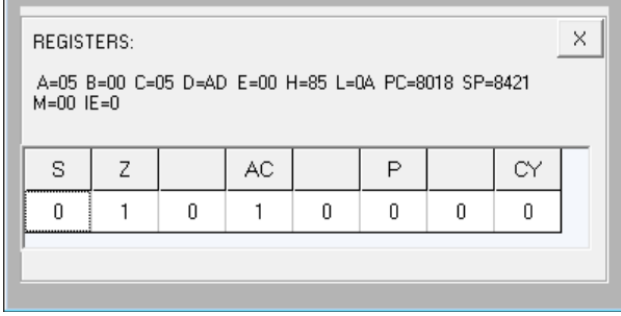
## **QUESTION NO: 12**

### **OBJECTIVE:**

Write an assembly language program to find the numbers of even parity in ten consecutive memory locations in 8085.

CODE	MEMORY LOCATION	OPCODE
MVI B, 0A	8000,8001	06,0A
MVI C, 00	8002,8003	0E,00
LXI H, 8500H	8004,8005,8006	21,00,85
Back: MOV A, M	8007	7E
ANI FF	8008,8009	E6,FF
JPO Next	800A,800B,800C	E2,0E,80
INR C	800D	0C
Next: INX H	800E	23
DCR B	800F	05
JNZ Back	8010,8011,8012	C2,07,80
MOV A, C	8013	79
STA 8600H	8014,8015,8016	32,00,86
RST 5	8017	EF

**Table 12.1:** Code Explanation



REGISTERS:							
A=05 B=00 C=05 D=AD E=00 H=85 L=0A PC=8018 SP=8421 M=00 IE=0							
S	Z		AC		P		CY
0	1	0	1	0	0	0	0

**Fig. 12.1:** Register Output

debugform			
ROW NUMBER		Show ROW	
ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	06	MVI B,8 bit	2
8001	0A		
8002	0E	MVI C,8 bit	2
8003	00		
8004	21	LXI H,16 bit	3
8005	00		
8006	85		
8007	7E	MOV A,M	1
8008	E6	ANI 8 bit	2
8009	FF		
800A	E2	JPD 16 bit	3
800B	0E		
800C	80		
JUMPED TO 800E			
800F	23	INX H	1
STACK (LIFO)			

**Fig. 12.2:** Debugform



**Fig. 12`.3:** Output

## INPUT –

[8500] – 01, [8501] – 03, [8502] – 01, [8503] – 03, [8504] – 01, [8505] – 03, [8506] – 01, [8507] – 03, [8508] – 01, [8509] – 03

## OUTPUT –

[8600] – 05

## **QUESTION NO: 13**

### **OBJECTIVE:**

Write an assembly language program to convert a BCD number into its equivalent binary in 8085.

<b>CODE</b>	<b>MEMORY LOCATION</b>	<b>OPCODE</b>
<b>LDA 8500H</b>	<b>8000,8001,8002</b>	<b>3A,00,85</b>
<b>MOV B, A</b>	<b>8003</b>	<b>47</b>
<b>ANI 0F</b>	<b>8004,8005</b>	<b>E6,0F</b>
<b>MOV C, A</b>	<b>8006</b>	<b>4F</b>
<b>MOV A, B</b>	<b>8007</b>	<b>78</b>
<b>ANI F0</b>	<b>8008,8009</b>	<b>E6,F0</b>
<b>RRC</b>	<b>800A</b>	<b>0F</b>
<b>RRC</b>	<b>800B</b>	<b>0F</b>
<b>RRC</b>	<b>800C</b>	<b>0F</b>
<b>RRC</b>	<b>800D</b>	<b>0F</b>
<b>MOV B, A</b>	<b>800E</b>	<b>47</b>
<b>XRA A</b>	<b>800F</b>	<b>AF</b>
<b>MVI D, 0A</b>	<b>8010,8011</b>	<b>16,0A</b>
<b>Sum: ADD D</b>	<b>8012</b>	<b>82</b>
<b>DCR B</b>	<b>8013</b>	<b>05</b>
<b>JNZ Sum</b>	<b>8014,8015,8016</b>	<b>C2,12,80</b>
<b>ADD C</b>	<b>8017</b>	<b>81</b>
<b>STA 8600H</b>	<b>8018,8019,801A</b>	<b>32,00,86</b>
<b>RST 5</b>	<b>801B</b>	<b>EF</b>

**Table 13.1:** Code Explanation

REGISTERS:							
A=43 B=00 C=07 D=0A E=00 H=85 L=0A PC=801C SP=8421 M=00 IE=0							
S	Z		AC		P		CY
0	0	0	1	0	0	0	0

**Fig. 13.1:** Register Output

debugform			
ROW NUMBER		Show ROW	
ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	3A	LDA 16 bit	3
8001	00		
8002	85		
8003	47	MOV B,A	1
8004	E6	ANI 8 bit	2
8005	0F		
8006	4F	MOV C,A	1
8007	78	MOV A,B	1
8008	E6	ANI 8 bit	2
8009	F0		
800A	0F	RRC	1
800B	0F	RRC	1
800C	0F	RRC	1
800D	0F	RRC	1
800E	47	MOV B,A	1
800F	AF	XRA A	1
STACK (LIFO)			

**Fig. 13.2:** Debugform

8085
8600
43

**Fig. 13.3:** Output

**INPUT –**

[8500] – 67

**OUTPUT –**

[8600] – 43

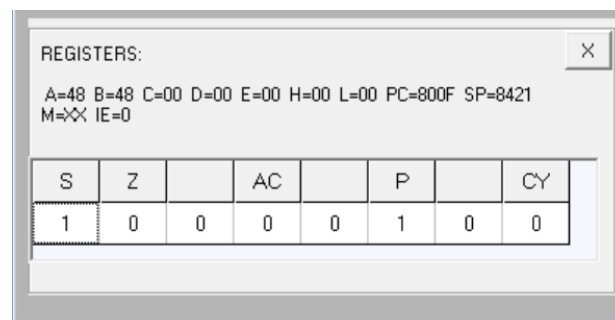
## QUESTION NO: 14

### OBJECTIVE:

Write an ALP for exchange the contents of memory location.

CODE	MEMORY LOCATION	OPCODE
LDA 8500H	8000, 8001, 8002	3A, 00, 85
MOV B, A	8003	47
LDA 8600H	8004,8005,8006	3A, 00, 86
STA 8500H	8007,8008,8009	32, 00, 85
MOV A, B	800A	78
STA 8600H	800B,800C,800D	32, 00, 86
RST 5	800E	EF

**Table 14.1:** Code Explanation



REGISTERS:

A=48 B=48 C=00 D=00 E=00 H=00 L=00 PC=800F SP=8421  
M=XX IE=0

S	Z		AC		P		CY
1	0	0	0	0	1	0	0

**Fig. 14.1:** Register Output

debugform			
ROW NUMBER		Show ROW	
ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	3A	LDA 16 bit	3
8001	00		
8002	85		
8003	47	MOV B,A	1
8004	3A	LDA 16 bit	3
8005	00		
8006	86		
8007	32	STA 16 bit	3
8008	00		
8009	85		
800A	78	MOV A,B	1
800B	32	STA 16 bit	3
800C	00		
800D	86		
800E	EF	RST 5	1
STACK (LIFO)			

**Fig. 14.2:** Debugform

8085	8500	88
8085	8600	48

**Fig. 14.3:** Output

**INPUT –**

[8500] – 48, [8600] – 88

**OUTPUT –**

[8500] – 88, [8600] – 48

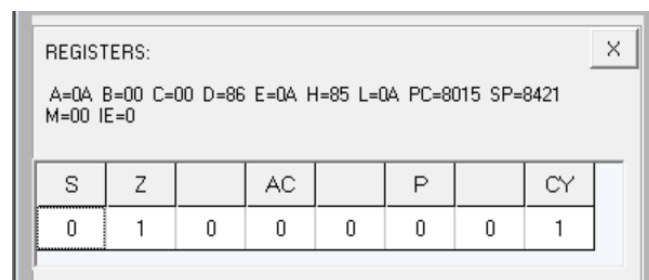
## **QUESTION NO: 15**

### **OBJECTIVE:**

Write a program to find the largest number in an array of 10 elements.

CODE	MEMORY LOCATION	OPCODE
<b>MVI B, 09</b>	8000,8001	06,09
<b>LXI H, 8500H</b>	8002,8003,8004	21,00,85
<b>MOV A, M</b>	8005	7E
<b>INX H</b>	8006	23
<b>Back: CMP M</b>	8007	BC
<b>JNC Next</b>	8008,8009,800A	D2,0C,80
<b>MOV A, M</b>	800B	7E
<b>Next: INX H</b>	800C	23
<b>DCR B</b>	800D	05
<b>JNZ Back</b>	800E,800F,8010	C2,07,80
<b>STA 850AH</b>	8011,8012,8013	32,0A,85
<b>RST 5</b>	8014	EF

**Table 15.1:** Code Explanation



REGISTERS:

A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8015 SP=8421  
M=00 IE=0

S	Z		AC		P		CY
0	1	0	0	0	0	0	1

**Fig. 15.1:** Register Output



debugform

50 ROW NUMBER Show ROW

ADDRESS	OPCODE	INSTRUCTION	BYTES
8000	06	MVI B,8 bit	2
8001	09		
8002	21	LXI H,16 bit	3
8003	00		
8004	85		
8005	7E	MOV A,M	1
8006	23	INX H	1
8007	BC	CMP H	1
8008	D2	JNC 16 bit	3
8009	0C		
800A	80		
800B	7E	MOV A,M	1
800C	23	INX H	1
800D	05	DCR B	1
800E	C2	JNZ 16 bit	3
800F	07		

STACK (LIFO)

**Fig. 15.2:** Debugform



**Fig. 15.3:** Output

**INPUT –**

[8500] – 01, [8501] – 02,.....[8509] – 0A

**OUTPUT –**

[850A] – 0A

**QUESTION NO: 16**

**OBJECTIVE:**

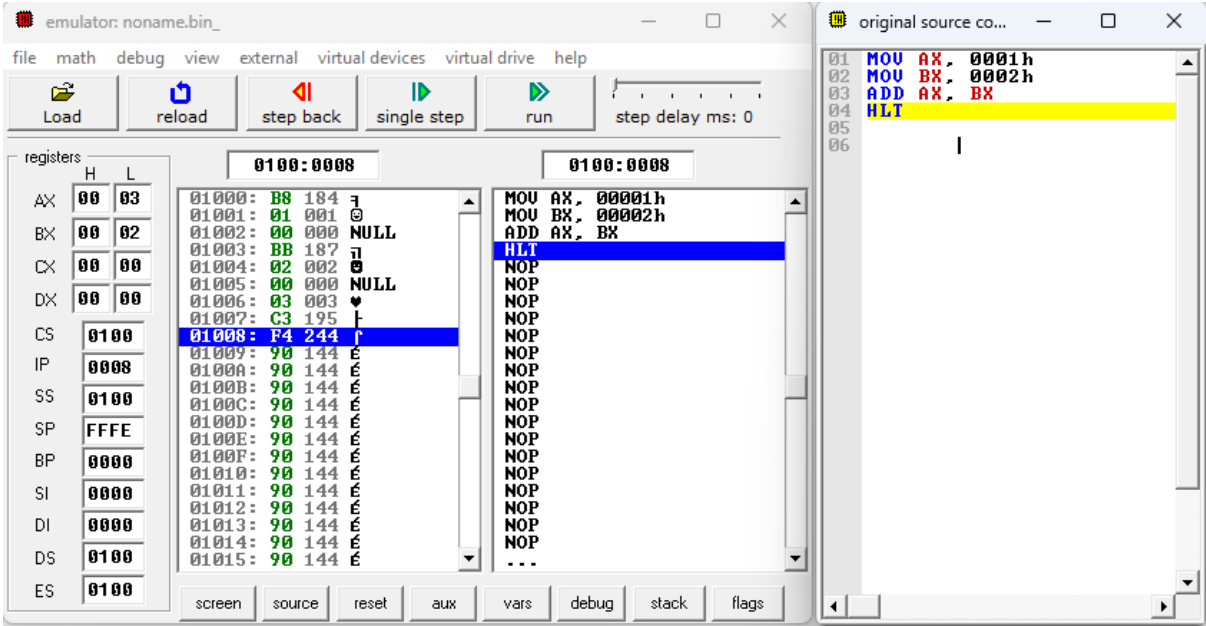
Write an assembly language program to add two 16-bit numbers in 8086.

MOV AX, 0001h

MOV BX, 0002h

ADD AX, BX

HLT



**Fig. 16.1:** Code Compilation and output

## OUTPUT –

**AX = 0003h**

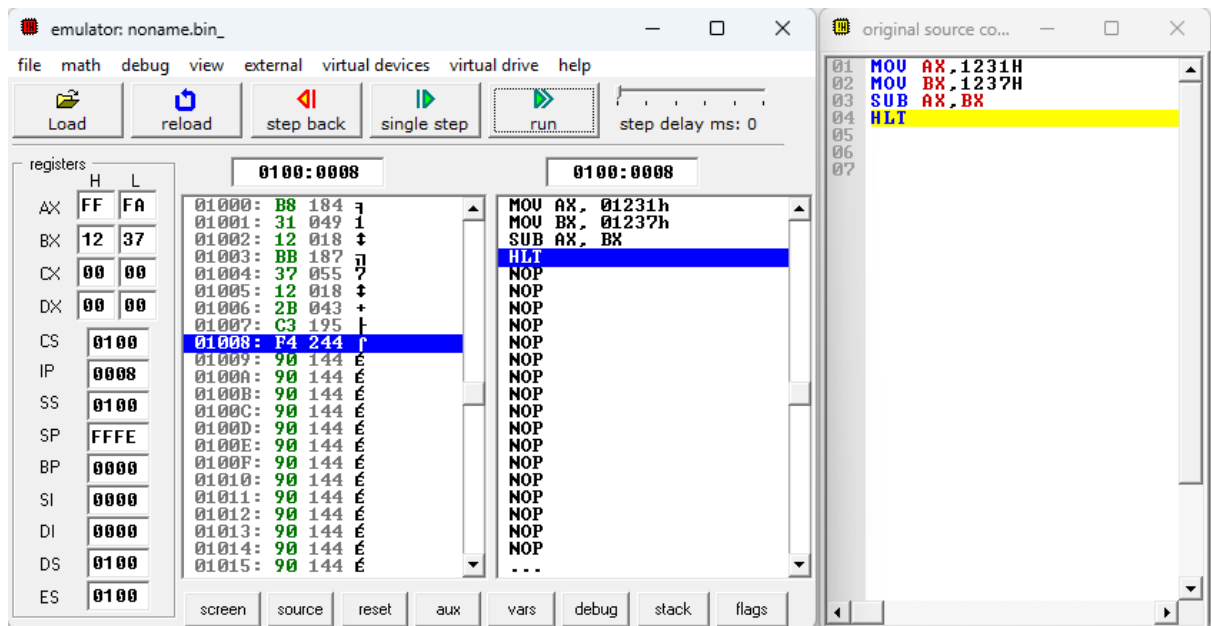
BX = 0002h

## QUESTION NO: 17

### OBJECTIVE:

Write an assembly language program to subtract two 16-bit numbers in 8086.

```
MOV AX,1231H
MOV BX,1237H
SUB AX,BX
HLT
```



**Fig. 17.1:** Code Compilation and output

### **OUTPUT –**

AX = FFFDh

BX = 1237h

## QUESTION NO: 18

### OBJECTIVE:

Write an assembly language program to multiply two 16-bit numbers in 8086.

```
MOV AX, 1234H
MOV BX, 56A4H
MUL BX
```

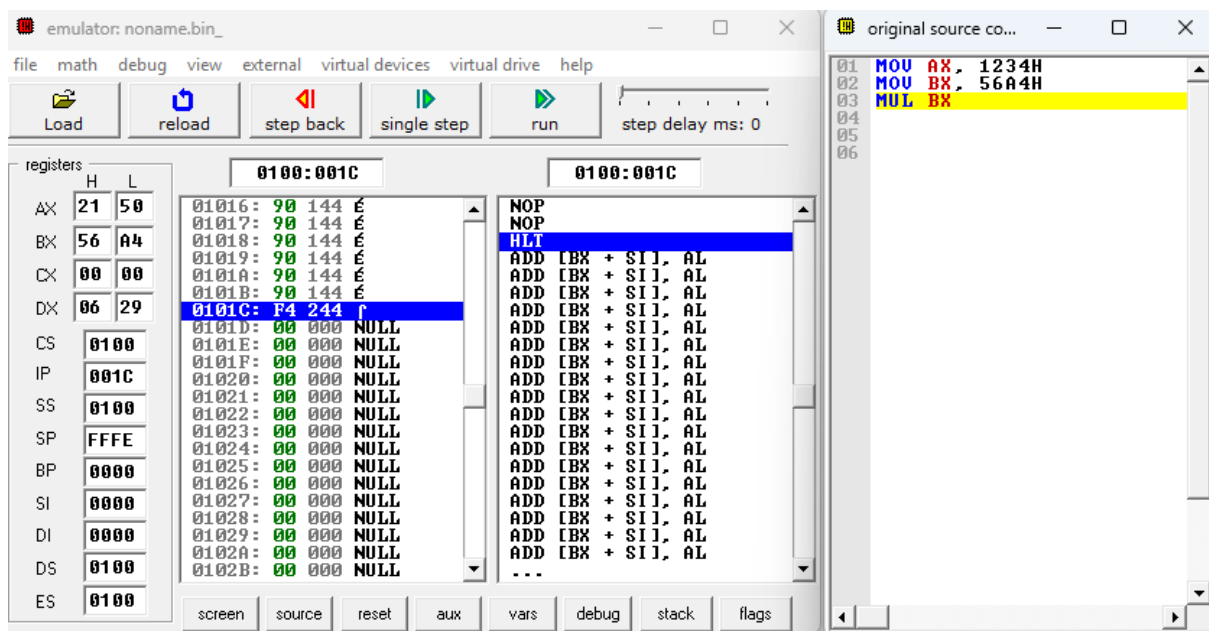


Fig. 18.1: Code Compilation and output

### **OUTPUT –**

AX = 2150h

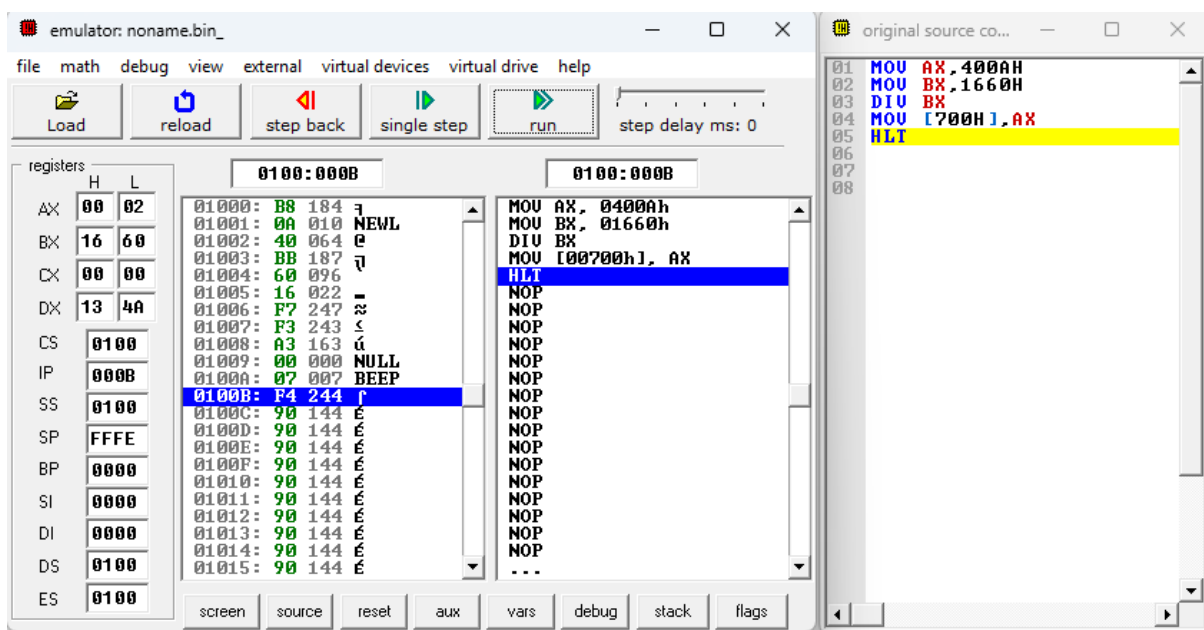
BX = 0629h

## QUESTION NO: 19

### OBJECTIVE:

Write an assembly language program to divide two 16-bit numbers in 8086.

```
MOV AX, 400AH
MOV BX, 1660H
DIV BX
MOV [700H], AX
HLT
```



**Fig. 19.1:** Code Compilation and output

### **OUTPUT –**

AX = 0002h (quotient)

DX = 134Ah (remainder)

## QUESTION NO: 20

### OBJECTIVE:

Write an assembly language program to demonstrate AAA, AAS, AAM, AAD, DAA and DAS in 8086.

#### i. AAA

```
MOV AL,'9'  
ADD AL,'2'  
AAA  
MOV [100H],AX  
HLT
```

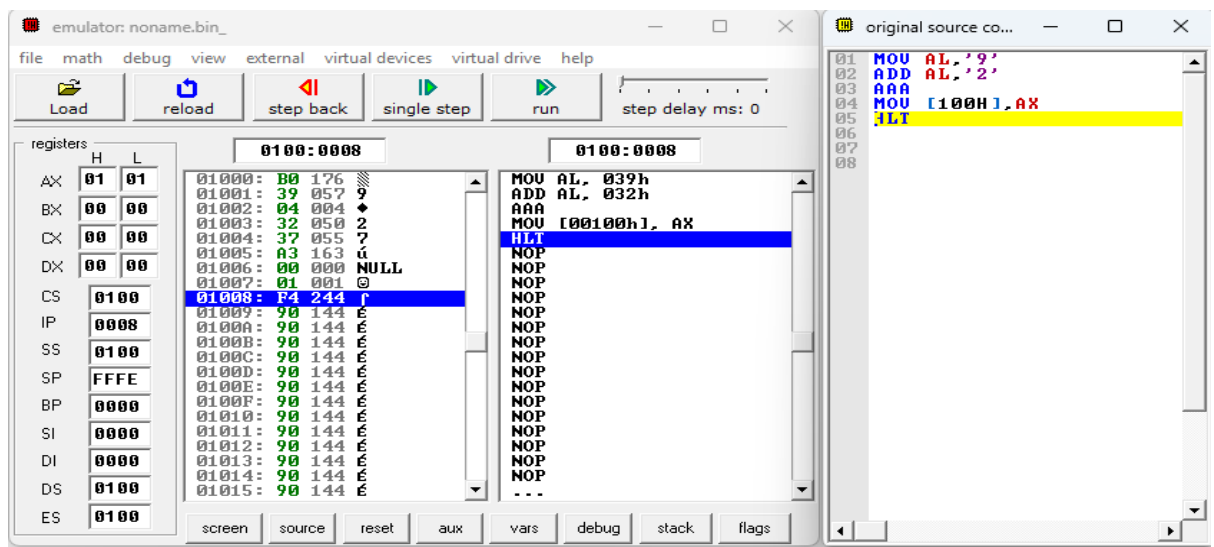


Fig. 20.1: Code Compilation and output

### OUTPUT –

AX = 0101h

#### ii. AAS

```
MOV AX,29H  
SUB AL,31H  
AAS  
ADD AX,2500H
```

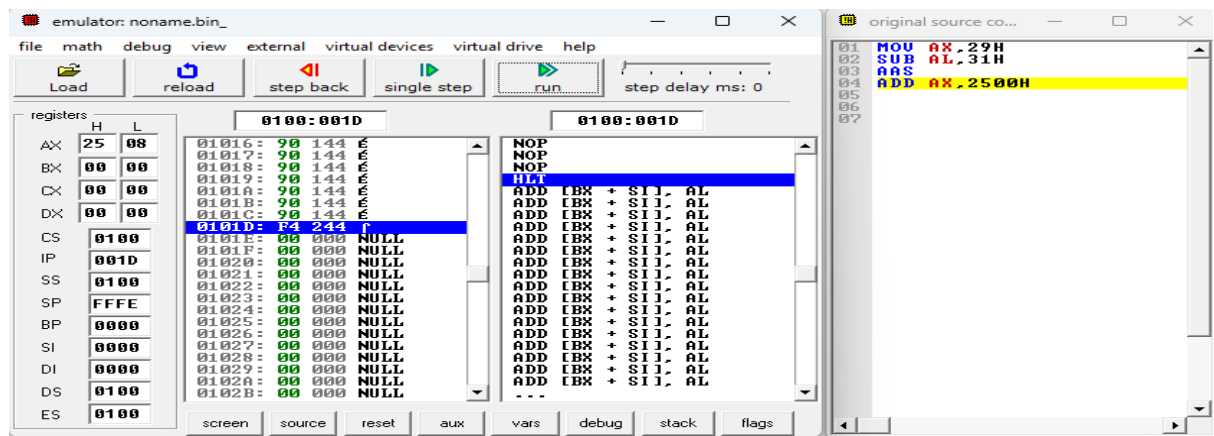


Fig. 20.2: Code Compilation and output

## OUTPUT –

AX = 2508h

### iii. AAM

MOV AL, 05H

MOV BL, 09H

MUL BL

AAM

MOV [100H], AX

HLT

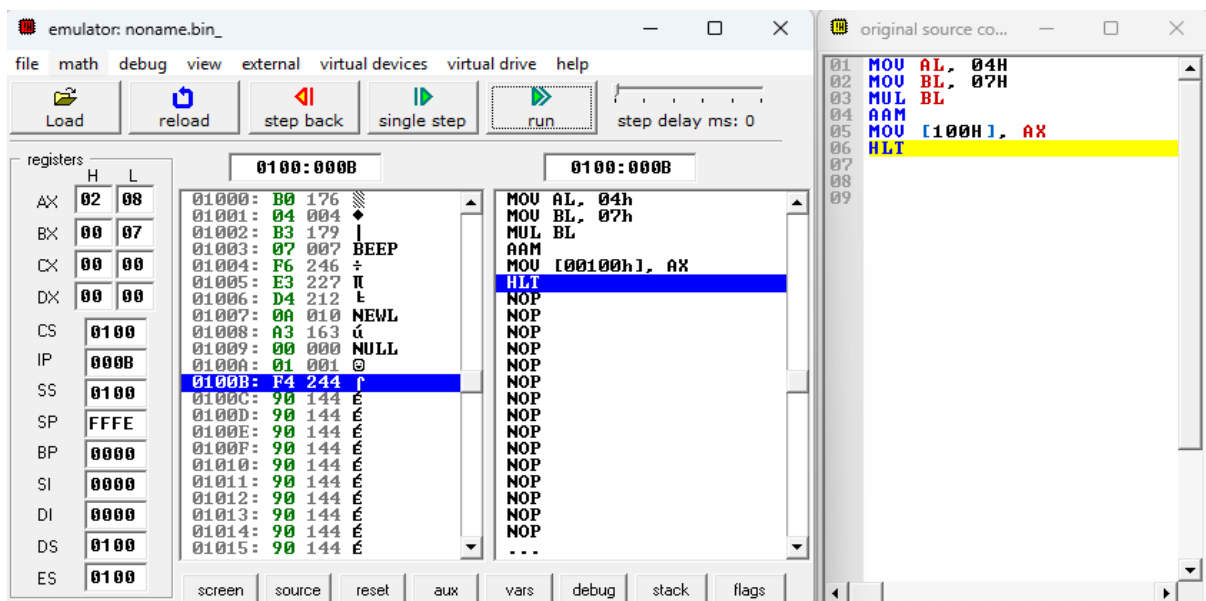


Fig. 20.3: Code Compilation and output

## OUTPUT –

AX = 2508h

### iv. AAD

```
MOV BL, 09H
MOV AX, 0607H
AAD
DIV BL
```

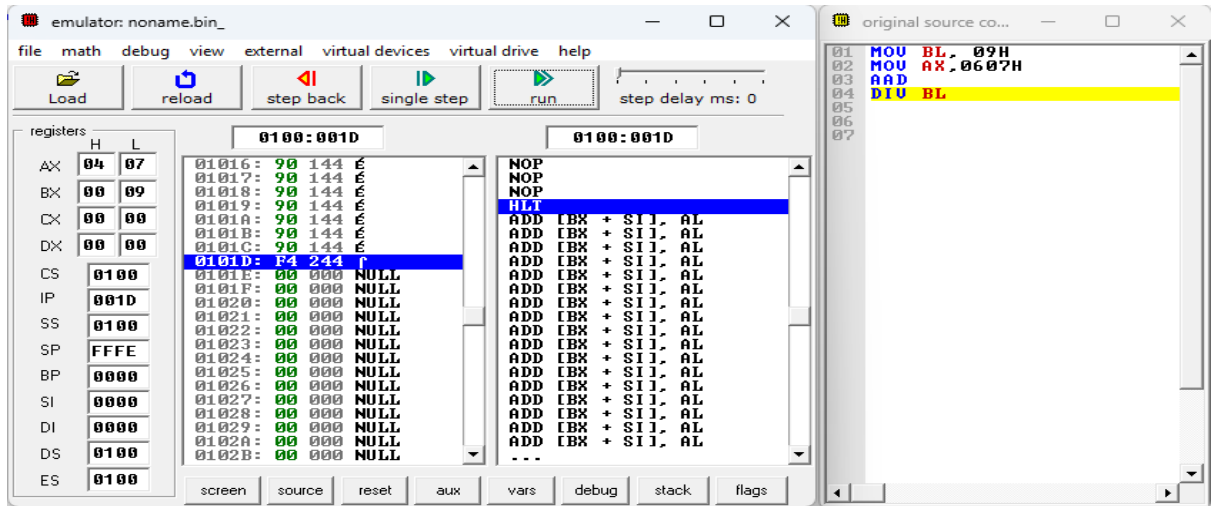


Fig. 20.4: Code Compilation and output

## OUTPUT –

AX = 0407h

### v. DAA

```
MOV AL, 9H
ADD AL, 1H
DAA
MOV [100H], AX
HLT
```

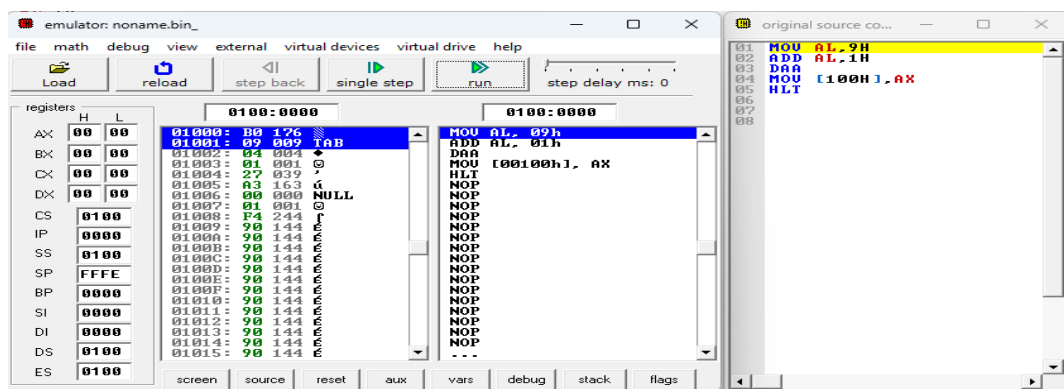


Fig. 20.5: Code Compilation and output



## vi. DAS

MOV AL,45H

SUB AL,27H

DAS

MOV [100H],AX

HLT

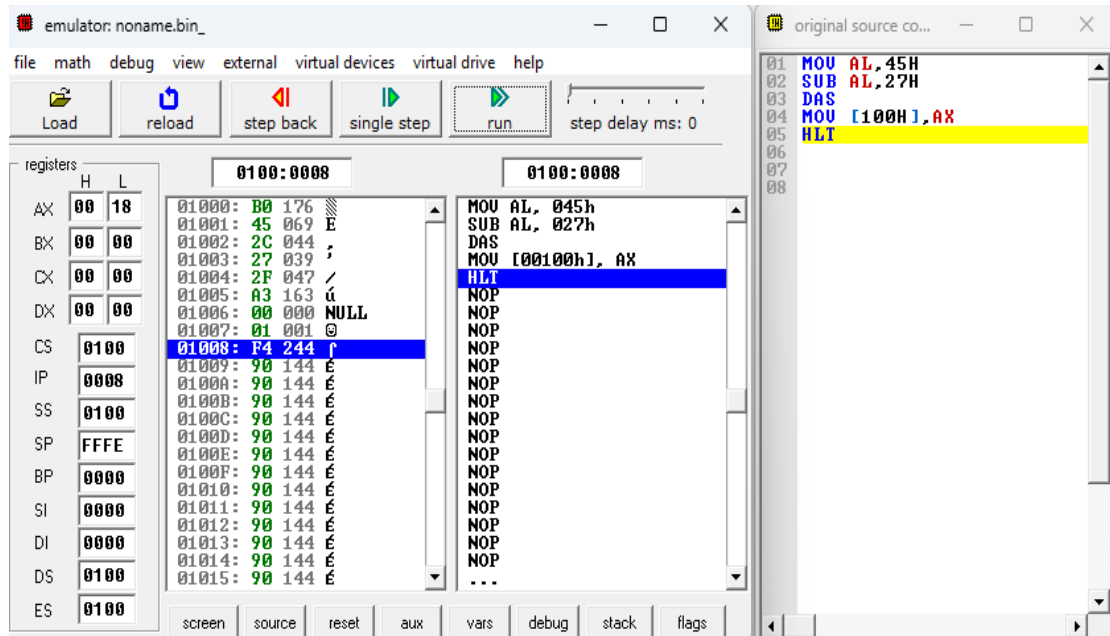


Fig. 20.6: Code Compilation and output

## OUTPUT –

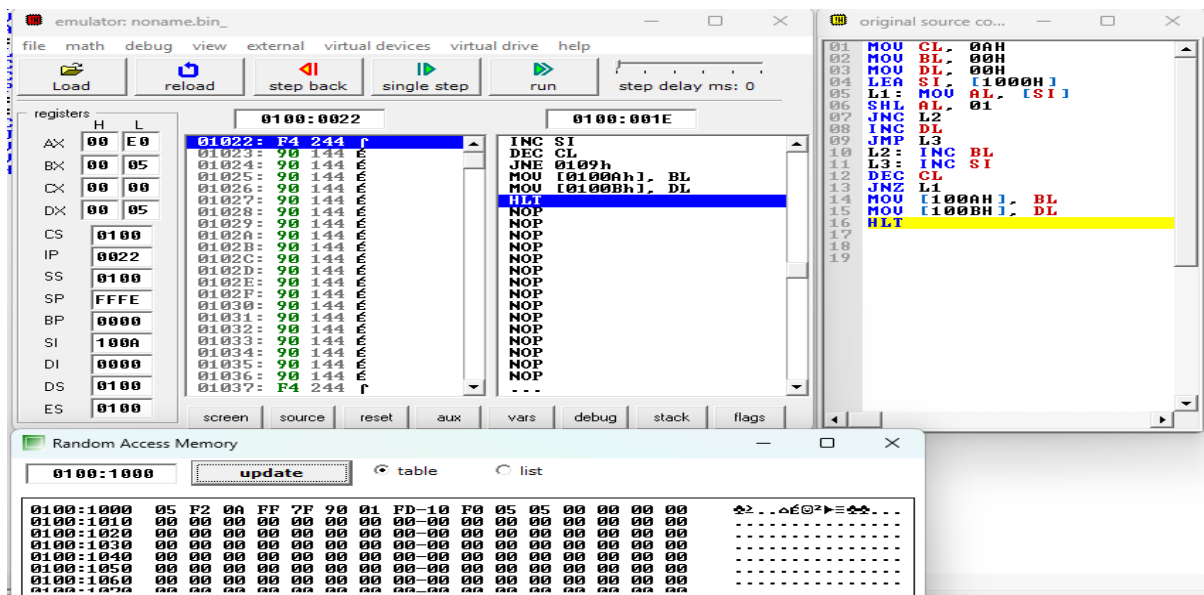
AX = 0018h

**QUESTION NO: 21**

**OBJECTIVE:**

Write an assembly language program to find out the count of positive numbers and negative numbers from a series of signed numbers in 8086.

```
MOV CL, 0AH
MOV BL, 00H
MOV DL, 00H
LEA SI, [1000H]
L1: MOV AL, [SI]
SHL AL, 01
JNC L2
INC DL
JMP L3
L2: INC BL
L3: INC SI
DEC CL
JNZ L1
MOV [100AH], BL
MOV [100BH], DL
HLT
```



**Fig. 21.1:** Code Compilation and output

**INPUT** -05, F2, 0A, FF, 7F, 90, 01, FD, 10, F0

**OUTPUT** – [100AH]=5 (positive)    [100BH]=5 (negative)

## QUESTION NO: 22

### OBJECTIVE:

Write an assembly language program to convert to find out the largest number from a given unordered array of 8-bit numbers, stored in the locations starting from a known address in 8086.

```
MOV CL, 0AH
LEA SI, [1000H]
MOV AL, [SI]
L1: INC SI
MOV BL, [SI]
CMP AL, BL
JC L2
JMP L3
L2: MOV AL, BL
L3: DEC CL
JNZ L1
MOV [100AH], AL
HLT
```

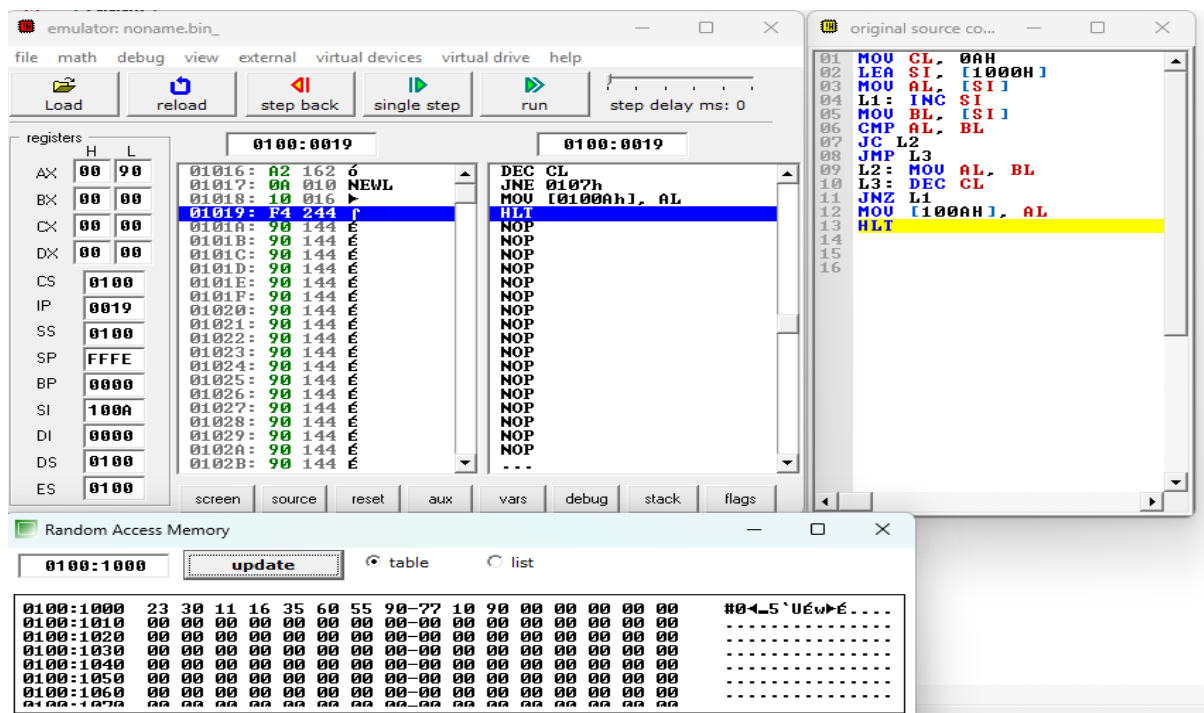


Fig. 22.1: Code Compilation and output

**INPUT-** 23, 30, 11, 16, 35, 60, 55, 90, 77, 10

**OUTPUT-** [100AH]= 90

## QUESTION NO: 23

### OBJECTIVE:

Write an assembly language program to convert to find out the largest number from a given unordered array of 16-bit numbers, stored in the locations starting from a known address in 8086.

```
LEA SI, [1200H]
MOV CL,[SI]
INC SI
MOV AX,[SI]
DEC CL
L2: INC SI
CMP AX,[SI]
JNB L1
MOV AL,[SI]
L1: DEC CL
JNZ L2
MOV DI,1300H
MOV [DI] , AX
HLT
```

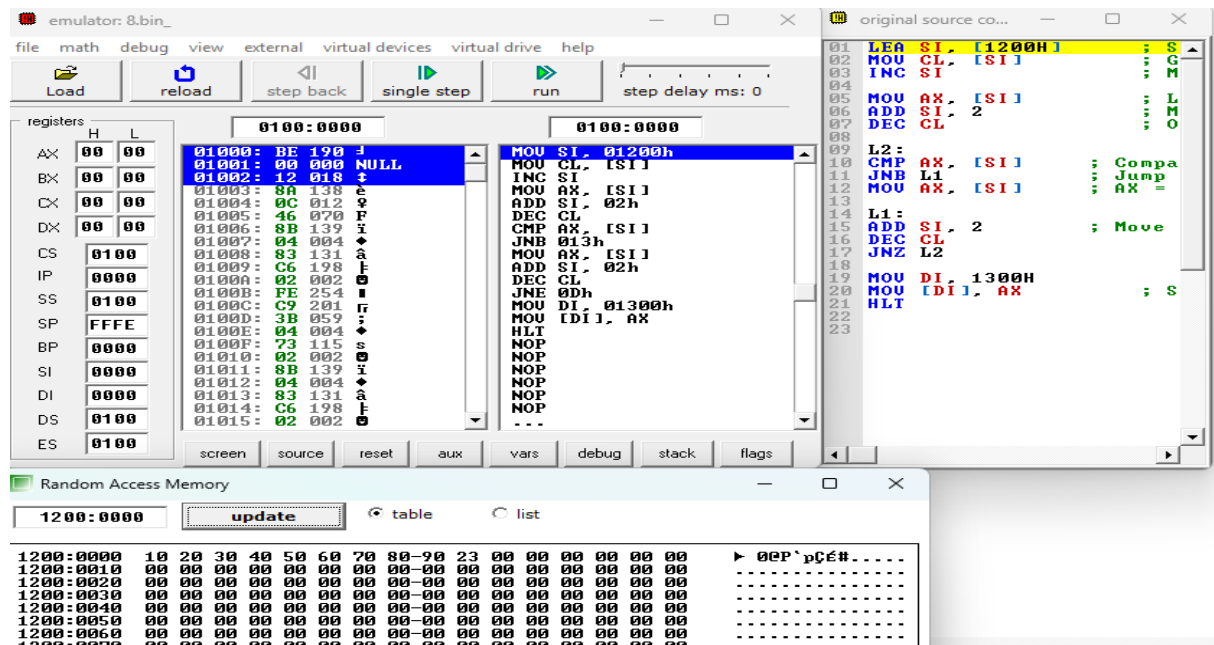


Fig. 23.1: Code Compilation and output

**INPUT-** 10, 20, 30, 40, 50, 60, 70, 80, 90, 23

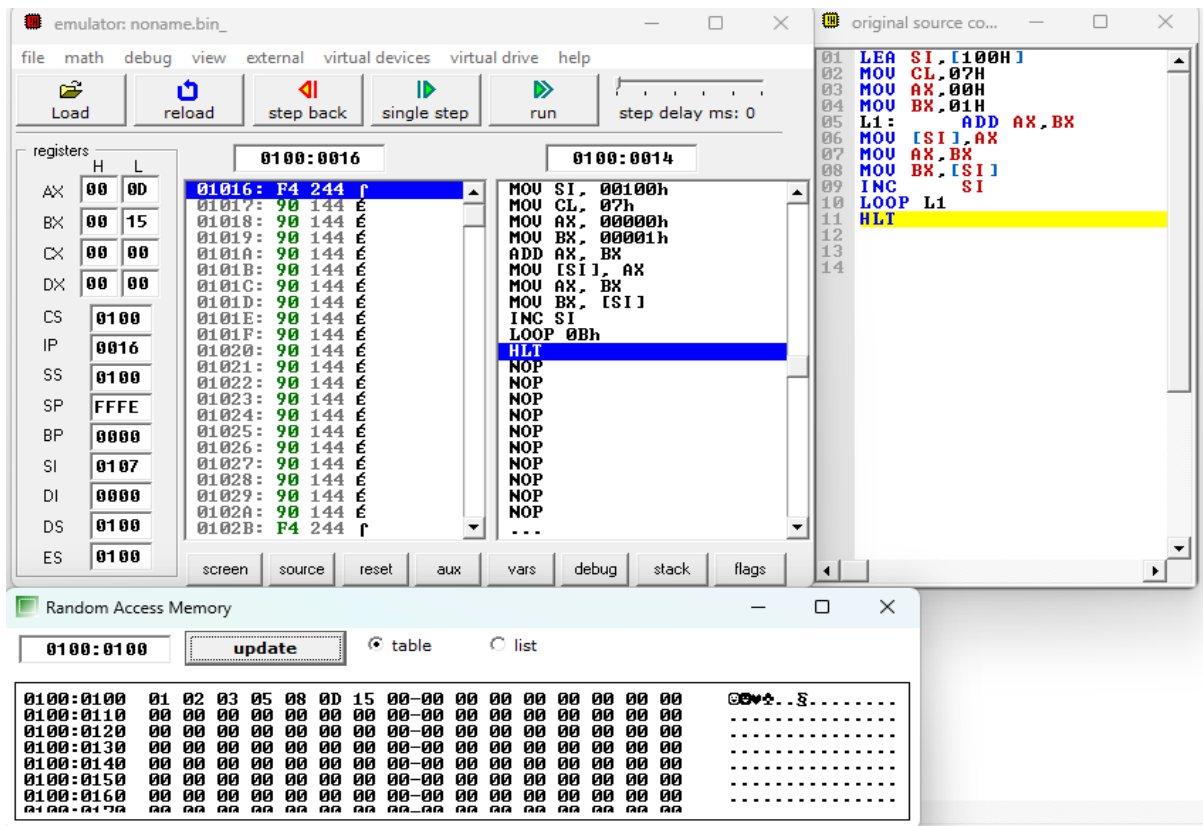
**OUTPUT-** 90

## QUESTION NO: 24

### OBJECTIVE:

Write an assembly language program to print Fibonacci series in 8086.

```
LEA SI,[100H]
MOV CL,07H
MOV AX,00H
MOV BX,01H
L1:  ADD AX, BX
    MOV [SI],AX
    MOV AX,BX
    MOV BX,[SI]
    INC  SI
    LOOP L1
    HLT
```



**Fig. 24.1:** Code Compilation and output

### **OUTPUT-**

[100H] = 00 (Fibonacci number 0), [101H] = 01 (Fibonacci number 1)

[102H] = 01 (Fibonacci number 1), [103H] = 02 (Fibonacci number 2)

[104H] = 03 (Fibonacci number 3), [105H] = 05 (Fibonacci number 5)

[106H] = 08 (Fibonacci number 8), [107H] = 13 (Fibonacci number 13)

## QUESTION NO: 25

### OBJECTIVE:

Write an assembly language program to perform the division 15/6 using the ASCII codes. Store the ASCII codes of the result in register DX.

```
MOV AX,'15'  
ADD BX,'6'  
SUB AX,3030H  
SUB BL,30H  
AAD  
DIV BL  
ADD AX,3030H  
MOV [100H],AX  
HLT
```

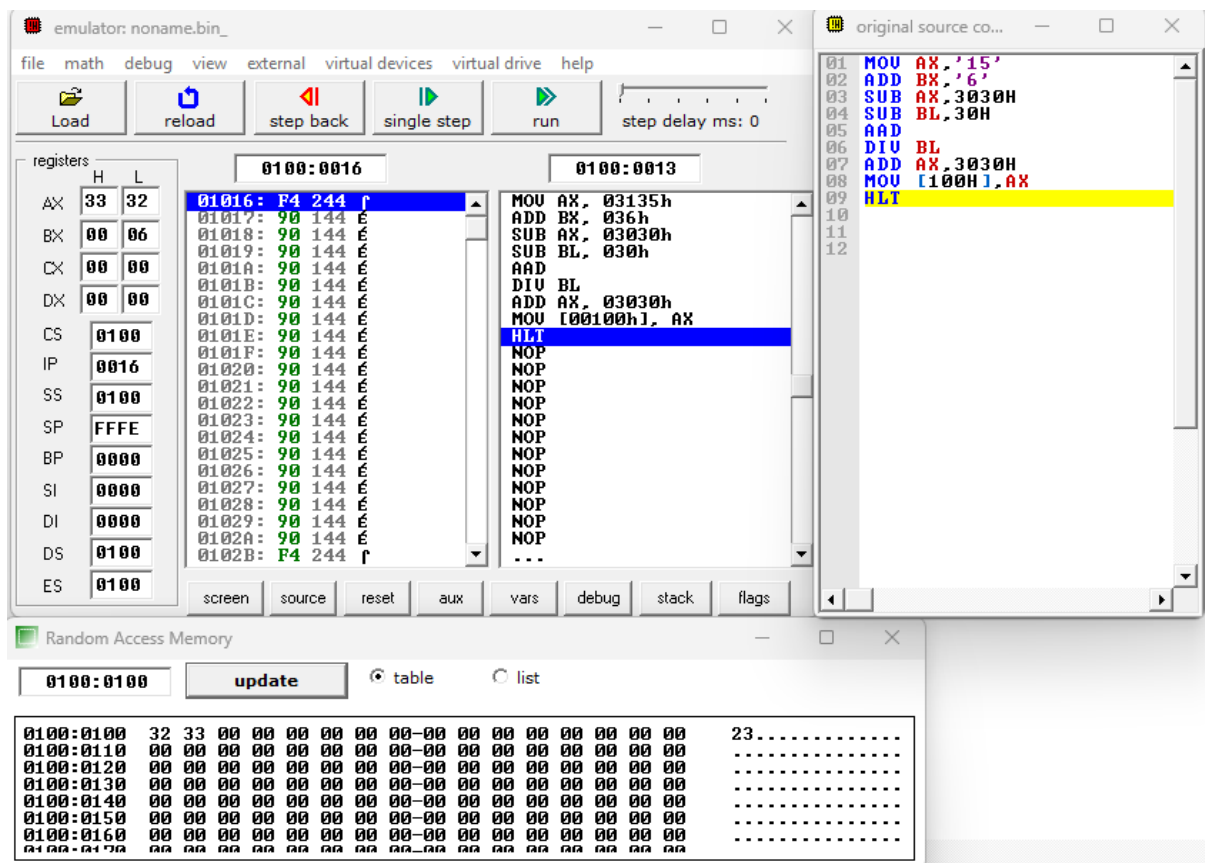


Fig. 25.1: Code Compilation and output

OUTPUT - AX= 3332

## **QUESTION NO: 26**

### **OBJECTIVE:**

Interfacing ET-8255 study card with ET-8086 microprocessor hardware kit.

### **SETUP FOR THE EXPERIMENT FOR 8086 KITS WITH LCD DISPLAY:**

**Step 1:-** Connect the card to the kit through 50-Pin cable. Ensure that the pin-1 of the card is connected to the pin-1 of the kit Bus connector, The Pin Nos. are marked in Kit and card.

**Step 2:-** Connect +5V and GND to the kit and switch on the supply. If the Kit used is with built in Power supply, then +5V and Gnd are internally connected and one just need to switch on the power and Press Reset.

**Step 3:-** Select the SW2 switch in OFF position & Enter the program given below using Sub\_MIr command if you want to enter in machine code or use EX\_MEM command for entering the program in Assembly language.

**Step 4:-** Execute the program using <GO> command. The Program stops after first I/O instruction. Observe the LED status as explained in the program below.

**Step 5:-** Press Sw1 one by one and observe the LED's status as explained in the program.

### **PROGRAM FOR INTERFACING 8255 STUDY CARD WITH 8086 KITS WITH LCD DISPLAY (i.e. ET-8086LCD and ET-8086ADLCD)**

CS:1P	OP-CODE	LABEL	MNEMONICS
1000:100	B080	START:	MOV AL,80
102	BA 86 00		MOV DX,86
105	EE		OUT DX, AL
106	B0 AA	LOOP:	MOV AL, AAH
108	BA 80 00		MOV DX,80
10B	EE		OUT DX, AL
10C	BA 82 00		MOV DX,82
10F	EE		OUT DX, AL

110	BA 84 00		MOV DX,84
113	EE		OUT DX, AL
114	B9 FF FF		MOV CX, FFFF
117	CD AA		INT AA
119	B0 55		MOV AL,55
11B	BA 80 00		MOV DX,80
11E	EE		OUT DX, AL
11F	BA 82 00		MOV DX,82
122	EE		OUT DX, AL
123	BA 84 00		MOV DX,84
126	EE		OUT DX, AL
127	B9 FF FF		MOV CX, FFFF
12A	CD AA		INT AA
12C	E8 D8		JMP 0106



## **QUESTION NO: 27**

### **OBJECTIVE:**

Interfacing ET-8253 study card with ET-8086 microprocessor hardware kit

### **SETUP FOR THE EXPERIMENT FOR ALL 8086 KITS WITH LCD DISPLAY:**

**Step 1:-** Connect the card to the kit through 50-Pin cable. Ensure that the pin-1 of the card is connected to the pin-1 of the kit Bus connector. The Pin Nos. are marked in Kit and card.

**Step 2:-** Connect +5V and GND to the kit and switch on the supply. If the Kit used is with built in Power supply, then +5V and Gnd are internally connected and one just need to switch on the power and Press Reset.

**Step 3:-** Select the SW2 switch in OFF position & Enter the program given below using EXMEM command.

**Step 4:-** Execute the program using <GO> command. The Program stops after the first I/O Instruction. Observe the LED status as explained in the program below.

**Step 5:-** Press SW1 one by one and observe the LED's status as explained in the observation. Observe square wave form on pin no. - 13 of 8253 or OUT1 Pin on the card.

### **PROGRAM FOR 8086 KIT WITH LCD DISPLAYS:- (i.e. ET-8086LCD, ET-8086ADLCD)**

CS:IP	OP-CODE	LABEL	MNEMONICS	COMMENTS	OBSERVATION
1001:100	BA 96 00	START:	MOV DX,96	Timer 1 is selected in mode 3, 16-bit binary counter.	Observe 76 on the data bus LEDs.  Observe A0, A1 LED ON, CS ON, WR* ON
0103	B0 76		MOV AL,76		
0105	EE		OUT DX, AL		
0106	B0 A0	XX:	MOV AL, A0	LOAD A0 as LSB first	

0108	BA 92 00		MOV DX,92		WR*, CS* ON, A0 & A1 OFF, WR* ON
10B	EE		OUT DX, AL		
10C	B0 00		MOV AL,00	LOAD A0 as MSB 00	Observe 76 on the data bus LEDs.  Observe A0
11E	EE		OUT DX, AL		AI LED ON, CS - ON, WR* -ON
11F	EB EF		JMP 0100	LOOP BACK	

## **QUESTION NO: 28**

### **OBJECTIVE:**

Interfacing ET-8279 study card with ET-8086 microprocessor hardware kit

### **SETUP FOR THE EXPERIMENT FOR 8086 KITS WITH LCD DISPLAY:**

**Step 1:-** Connect the card to the kit through a 50-Pin cable. Ensure that pin 1 of the card is connected to pin 1 of the kit bus connector. The pin numbers are marked on both the kit and card.

**Step 2:-** Connect +5V and GND to the kit and switch on the supply. If the kit has a built-in power supply, then +5V and GND are internally connected, and you just need to switch on the power and press reset.

**Step 3:-** Set the SW1 switch to the OFF position and enter the program given below using the EXMEM command.

**Step 4:-** Execute the program using the <GO> command. The program stops after the first I/O instruction. Observe the LED status as explained in the program below.

**Step 5:-** Press SW1 one by one and observe the LED's status as explained in the program. The program is in a loop after a while.

**Step 6:** Now short the Scan Line with the return line with a jumper wire provided e.g, RL2 with SL2. The INT LED will glow. Press the INT Key on the Board to allow the INT to reach the CPU.

**Step 7:** Press SW1 one by one and observe that the code of the key corresponding to RL2 and SL2 is displayed on the Kit as explained in the program.

**Step 8:** Repeat the process by shorting some other scan and Return lines. E.g. RL0 and SL0 and again note the code displayed.

### **PROGRAM FOR 8279 CARD FOR KEYBOARD WITH 8086 KITS WITH LCD DISPLAYS (i.e. ET-8086LCD, ET-8086ADLCD)**

ADDRESS	OP-CODE	LABEL	MNEMONICS
1000:100	B8 0F 11	START:	MOV AX, 110F
103	89 C4		MOV SP, AX
105	0E		PUSH CS

106	58		POP AX
107	89 C1		MOV CX, AX
109	BE 55 01		MOV SI, 0155
10C	B0 91		MOV AL, 91
10E	CD BE		INT BE
110	B0 13		MOV AL,13
112	BA 00 94		MOV DX, 9400
115	EE		OUT DX, AL
116	B0 90		MOV AL, 90
118	BA 02 94		MOV DX, 9402
11B	EE		OUT DX, AL
11C	B0 01		MOV AL, 01
11E	EE		OUT DX, AL
11F	B0 FD		MOV AL, FD
121	EE		OUT DX, AL
122	BA 72 01		MOV XD, 0172
125	B0 03		MOV AL,03

## **QUESTION NO: 29**

### **OBJECTIVE:**

Interfacing ET-8251 study card with ET-8086 microprocessor hardware kit.

### **SETUP FOR THE EXPERIMENT FOR ALL 8086 KITS WITH LCD DISPLAY:**

**Step 1:-** Connect the card to the kit through a 50-Pin cable. Ensure that pin-1 is connected to pin-1 of the kit Bus connector (the pin numbers are marked).

**Step 2:-** Connect +5V, +12V (Green wire), -12V (White wire), and GND to the Study Card, either via a separate power supply or a connector cable from the 8086 Kit to the four pin connector of the Study card (Only advanced 8086 kits have built-in +5V, +12V, and -12V supply; basic kits only have +5V — so extra supply is needed). Switch ON the power supply and press RESET on the kit.

**Step 3:-** Enter the sample program starting from address 1000:0100 using the appropriate command.

**Step 4:-** Execute the program using the <GO> key as explained in the kit manual.

**Step 5:-** Run PC Emulation software (like XTALK or PC LINK) on the PC. Set the baud rate to 2400.

**Step 6:-** Connect the serial cable from the study card to Port 1 of the PC.

**Step 7:-** Execute the program and observe the results on the LED's: Pressing the SW1 button sends the character "A" to the PC screen, each time you press SW1, it transmits "A".

### **PROGRAM FOR INTERFACING 8251 STUDY CARD WITH 8086 KITS WITH LCD DISPLAYS (i.e. ET-8086LCD, ET-8086ADLCD)**

ADDRESS	OP-CODE	COMMENT S	MNEMONICS
1000:0100	BA 82 00	Initialize 8251	MOV DX, 82
0103	B0 CE		MOV AL, CE
0105	EE		OUT DX, AL
0106	B0 37		MOV AL, 37

0108	EE		OUT DX, AL
0109	EC	Check for Tx flag	IN AL, DX
010A	24 01		AND AL, 01
010C	74 FB	If not check again	JZ 109
010E	B0 41	Load the ASCII Code of "A"	MOV AL, 41
0110	BA 80 00		MOV DX, 80
0113	EE		OUT DX, AL
0114	EB F8	Loop Back to out the next character	JMP 10E

On executing the above program the system will transmit character A repeatedly on the screen of the PC.

## **QUESTION NO: 30**

### **OBJECTIVE:**

Interfacing ET-8259 study card with ET-8086 microprocessor hardware kit.

### **SETUP FOR THE EXPERIMENT FOR 8086 KITS WITH LCD DISPLAY:**

In the experiments, the 8259 is used in stand alone mode, 8259 is programmed for level triggered, with call address interval of 4, No ICW4 required.

**Step 1:-** Connect the card to the kit through 50-Pin cable. Ensure that the pin-1 of the card is connected to the pin-1 of the kit Bus connector. The Pin Nos. are marked in Kit and the card.

**Step 2:-** Connect +5V and GND to the Kit and switch on the supply. If the Kit used is with in built power supply, then +5V and GND are internally connected and one just need to switch on the power and Press Reset.

**Step 3:-** Select the SW2 switch in OFF position and enter the program given below using EXMEM command.

**Step 4:-** Execute the program using <GO> command. The program stops after first I/O instruction. Observe the LED status as explained in the program below.

**Step 5:-** Press the single I/O instruction switch SW1 lever and observe the results on LED's till you reach the last instruction. You would have by now seen the programming. Now switch the SW2 to ON position. The program is now running in the full speed mode.

**Step 6:-** Now short the IR0 pin on the board to +5V Pin. You will see that the message SUPERB starts flashing on the kit.

### **PROGRAM FOR INTERFACING 8259 STUDY CARD WITH 8086 KIT WITH LCD DISPLAY (i.e. ET-8086LCD, ET-8086ADLCD)**

ADDRESS	OP-CODE	LABEL	MNEMONICS
1000:100	B8 0F 11	START:	MOV AX, 110F
103	89 C4		MOV SP, AX
105	0E		PUSH CS
106	58		POP AX
107	89 C1		MOV CX, AX

109	BE 55 01		MOV SI, 0155
10C	B0 91		MOV AL, 91
10E	CD BE		INT BE
110	BA 80 00		MOV DX, 80
113	B0 19		MOV AL, 19
115	EE		OUT DX, AL
116	B0 90		MOV AL, 90
118	BA 82 00		MOV DX, 82
11B	EE		OUT DX, AL
11C	B0 01		MOV AL, 01
11E	EE		OUT DX, AL
11F	B0 FD		MOV AL, FD
121	EE		OUT DX, AL
122	FB		ST1
123	EB FE	HERE:	JMP 123

**Now enter the following program at the address 1000:0155**

ADDRESS	OP-CODE	LABEL	MNEMONIC
1000:0155	CD AC	SER_ROUTINE:	INT AC
157	BA 41 00	AG86:	MOV DX, 41
15A	B4 02		MOV AH, 02
15C	CD A2		INT A2
15E	CD AB		INT AB
160	EB F5		JMP 0157