ASSIGNMENTS

Microprocessor Based Systems Design (UCS617)

Lab Assignment-1 (8085)

Submitted by:

Aditya Sayal 102103559

Manjot Singh Selopal 102103560

Mannat Sadana 102103561

Samarjeet Singh 102103562

Arshiya Kishore 102103565

Submitted to

Dr.Manju Khurana

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY,

(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB

INDIA

Jan-May 2024

INDEX

Sr. no.	Title	Page no.
1.	Introduction of 8085-microprocessor kit and steps for execution on the kit.	1-6
2.	Familiarity with 8085-microprocessor kit.	
2.1	Write a program to store 8-bit data into one register and then copy that to all registers.	7-9
2.2	Write a program for addition of two 8-bit numbers.	10-11
2.3	Write a program to add 8-bit numbers using direct and indirect addressing mode.	12-14
2.4	Write a program to add 16-bit numbers using direct and indirect addressing mode.	15-19
2.5	Write a program to 8-bit numbers using carry. (using JNC instruction).	20-22
2.6	Write a program to find 1’s complement and 2’s complement of 8-bit numbers.	23-26
3	Write a program for the sum of a series of numbers.	27-28
4	Write a program for data transfer from memory block B1 to memory block B2.	29-35
5	Write a program for multiplying two 8-bit numbers.	36-38
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 addresses.	39-40
7.	Write a program to find the negative numbers in a block of data.	41-42
8.	Write a program to count the number of one's in a number.	43-44
9.	Write a program to arrange numbers in Ascending order.	45-46
10.	Calculate the sum of a series of even numbers.	47-48

11.	Write an assembly language program to verify how many bytes are present in a given set , which resembles 10101101 in 8085.	49-50
12.	Write an assembly language program to find the numbers of even parity in ten consecutive memory locations in 8085.	51-52
13.	Write an assembly language program to convert a BCD number into its equivalent binary in 8085.	53-54
14.	Write an assembly language program for exchanging the contents of memory location.	55-56
15.	Write a program to find the largest number in an array of 10 elements.	57-58

Experiment No.-1

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

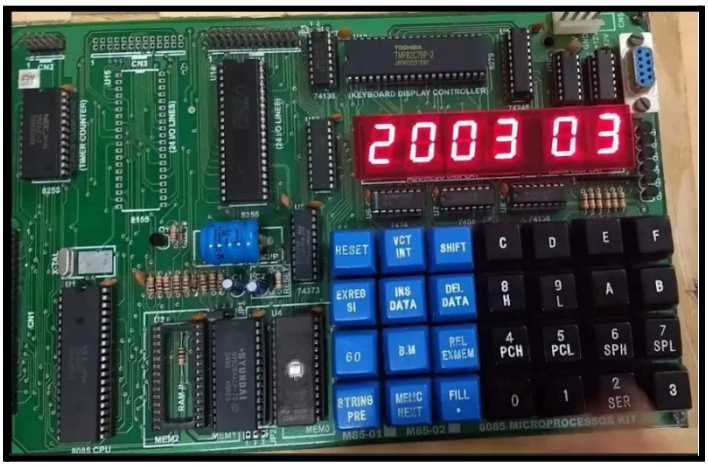

Fig 1.1

Microprocessor is a hardware component of a computer, and it works as the brain of the computer system. Microprocessor is a control unit of a computer because it is able to manage various Arithmetic Logical Unit (ALU) operations.

Features of Microprocessor- 8085

- 8085 is developed by INTEL.
- 8 bit microprocessor: can accept 8 bit data simultaneously.
- Operates on single +5V D.C. supply.
- Designed using NMOS technology.
- 6200 transistor on a single chip.
- It provides an on chip clock generator, hence it does not require an external clock generator.
- Operates on 3MHz clock frequency.
- 8-bit multiplexed address/data bus, which reduces the number of pins.
- 16-bit address lines, hence it can address 2^16 = 64 K bytes of memory
- 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
- It provides DMA (Direct memory access).
- 40-pin I.C. package fabricated on a single LSI chip.
- Clock cycle is 320ns.
- 80 basic instructions and 246 opcodes.

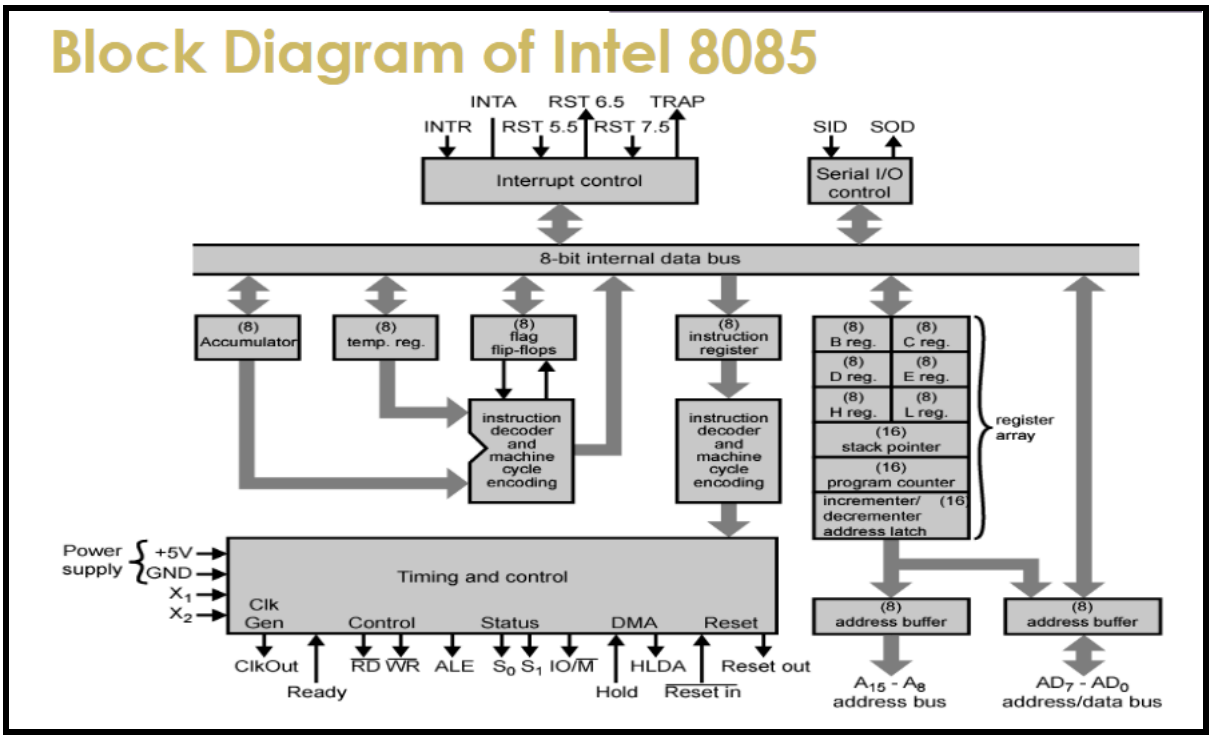

Fig 1.2

Intel 8085 Architecture

1. Registers Array

- General purpose register : (user accessible)
- B, C, D, E, H, L are 8 bit registers. (can be used singly)
- H & L can be used as a data pointer (holds memory address)
- Special Purpose Register [A, Instruction Register and Flag]
  Accumulator (A) : (user accessible)-8 bit register
  Instruction Register: (user not accessible)
  Flag Register (F) : (user accessible)-8 bit Register

2. ALU & Logical Group

- It consists of ALU, Accumulator, Temporary register & Flag Register
- ALU performs arithmetic and logical operations.
- Accumulator-General purpose register.
- Temporary Register-8 bit register, during the arithmetic and logical operations one operand is available in A and other operand is always transferred to temporary register.
- Flag Register-Five flags are connected to ALU

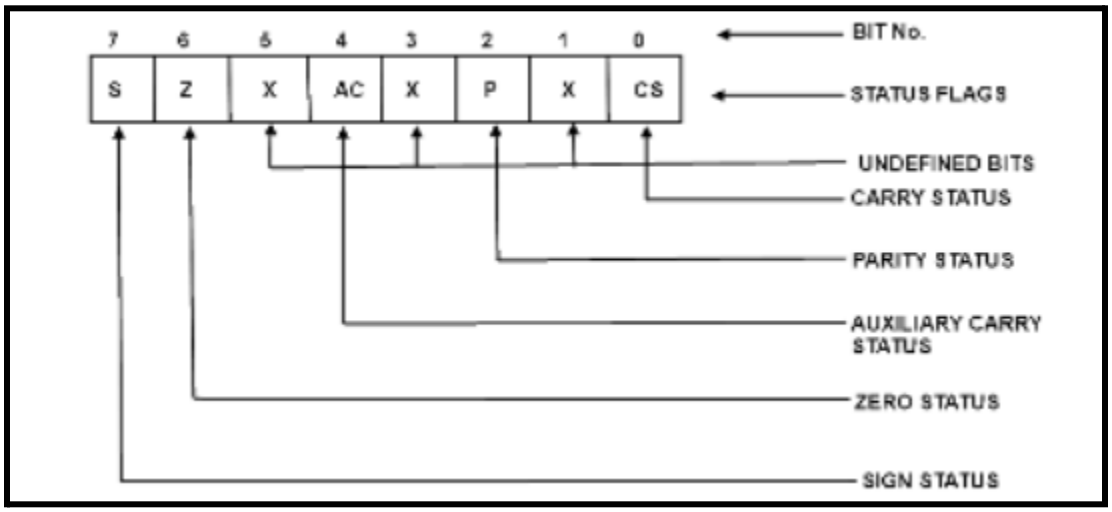

Fig 1.3

3. Instruction decoder and machine cycle encoder, Timing and control circuitry

- Instruction decoder and machine cycle decoder decodes the op-code stored in the Instruction Register (IR) and establishes the sequence of events to follow and encodes it and transfers to the timing & control unit to perform the execution of the instruction.
- Timing and control circuitry works as the brain of the CPU for proper sequence and synchronization of all the operations.

4. Interrupt Control group

- Interrupt:- Occurrence of an external disturbance
- After servicing the interrupt, 8085 resumes its normal working sequence
- Transfer the control to special routines
- Five interrupts: - TRAP, RST7.5, RST6.5, RST5.5, INTR
- In response to INTR, it generates INTA signal

5. Serial I/O control Group

- Data transfer D0- D7 lines is parallel data
- Serial data is entered through SID (serial input data) input (received)
- Serial data is outputted on SOD (serial output data) input (send)

Vikas Simulator Screenshot

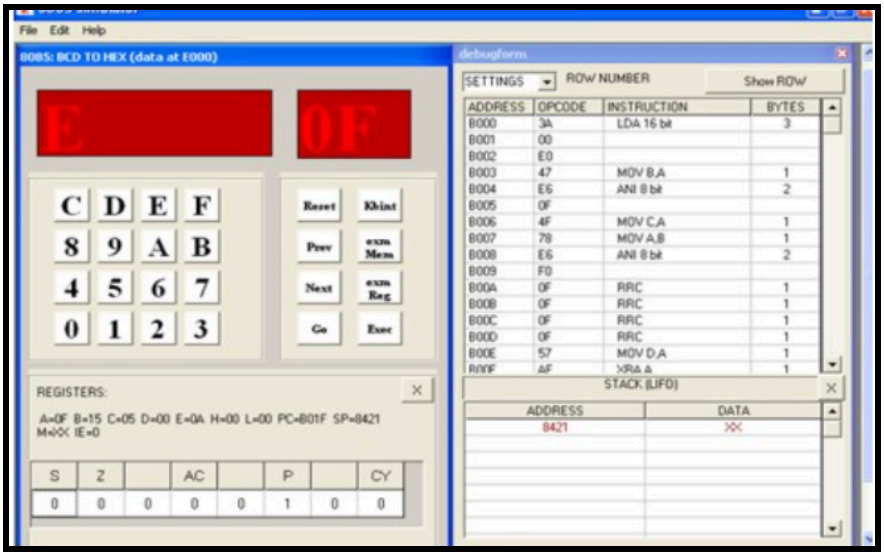

Fig 1.4

Steps to perform on the Intel kit as well as on Simulator

- Press Reset
- Press Examine Memory
- Enter starting address
- Press Next
- Enter opcodes by subsequently pressing Next
- Press Reset
- Press Go
- Enter starting address of the program to compile
- Press EXEC/FILL
- Press Reset
- Press Examine Memory
- Enter Output Address
- Press Next

Experiment No.-2

Familiarity with 8085 microprocessor kit.

Aim: 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 2.1

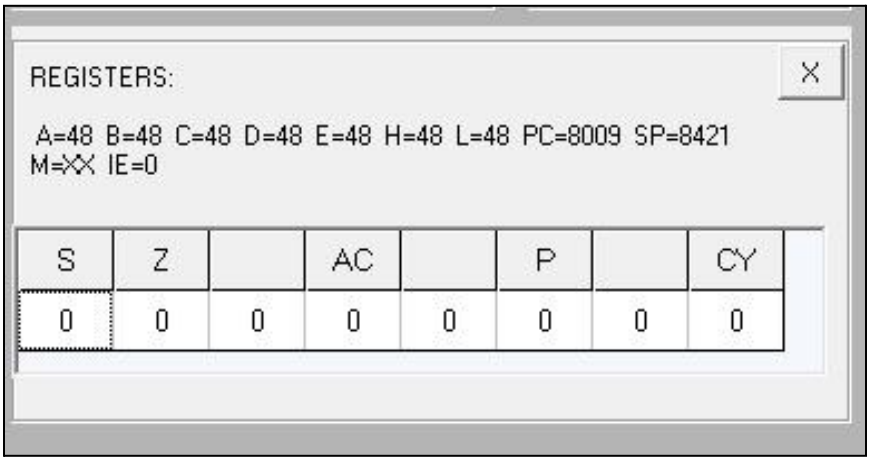

Fig 2.1

Output –

A – 48, B – 48, C – 48, D – 48, E – 48, H – 48, L – 48

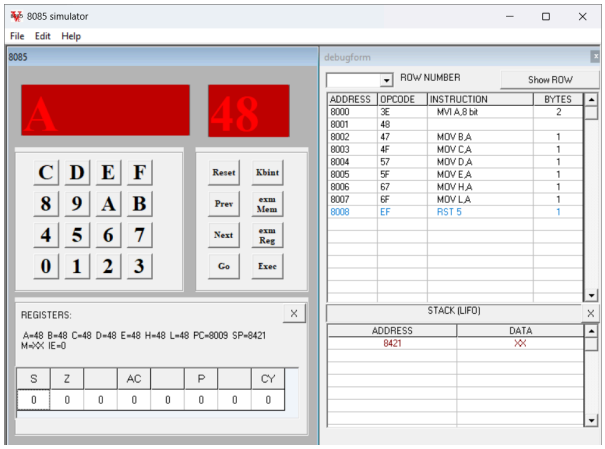

Fig 2.2

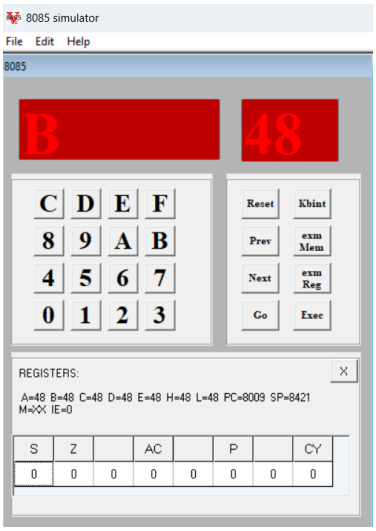

Fig 2.3

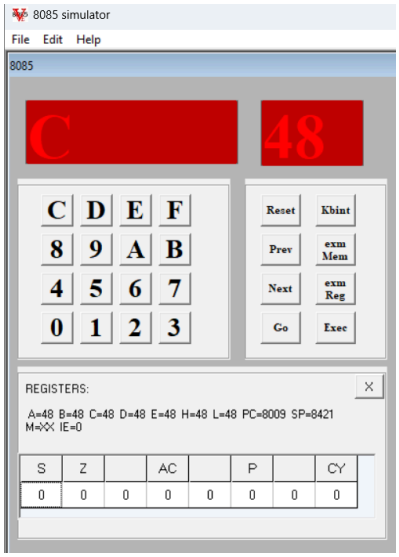

Fig 2.4

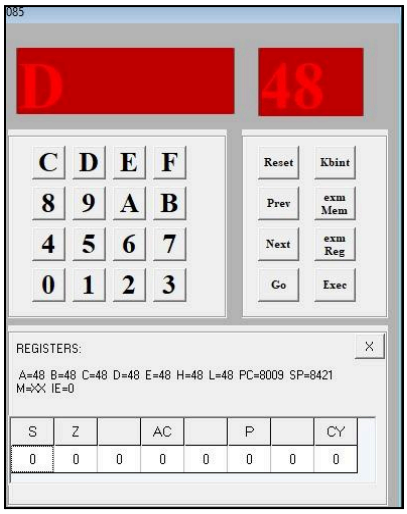

Fig 2.5

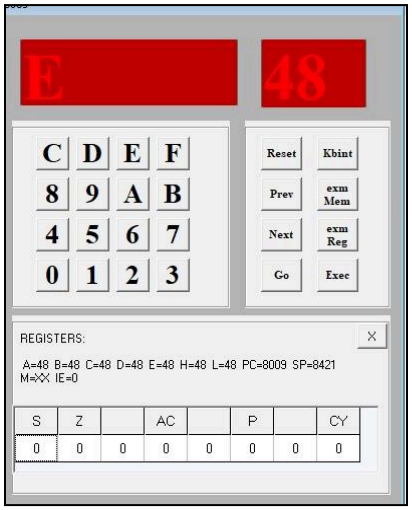

Fig 2.6

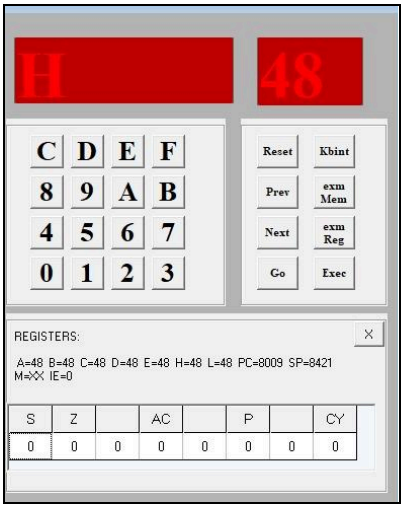

Fig 2.7

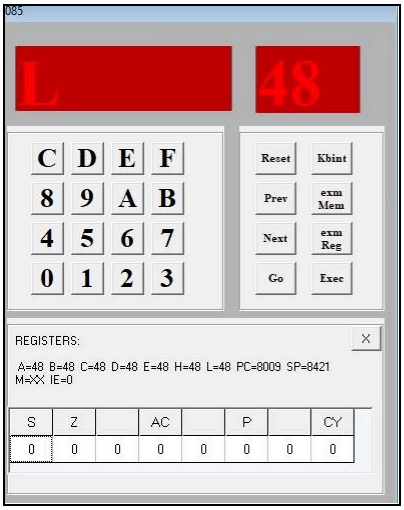

Fig 2.8

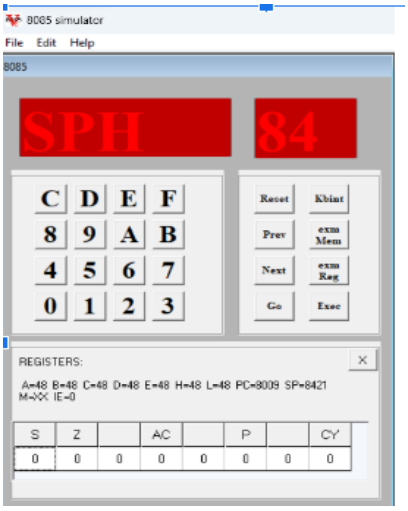

Fig 2.9

Fig 2.10

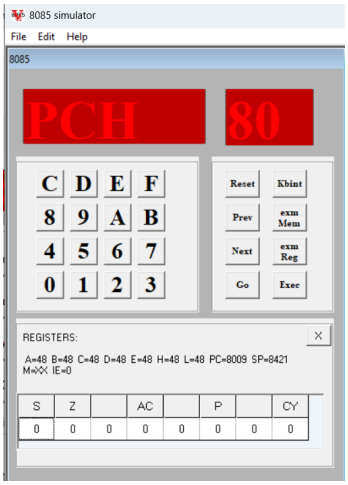

Fig 2.11

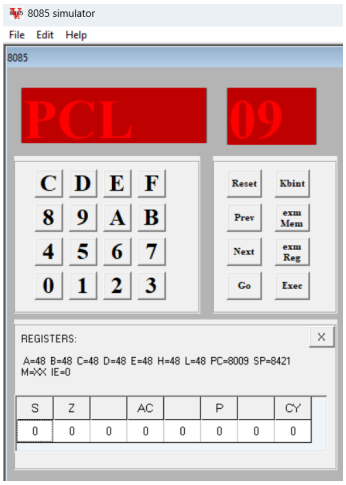

Fig2.12

Program 2.2

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 2.2

Output –

[8500] – 90

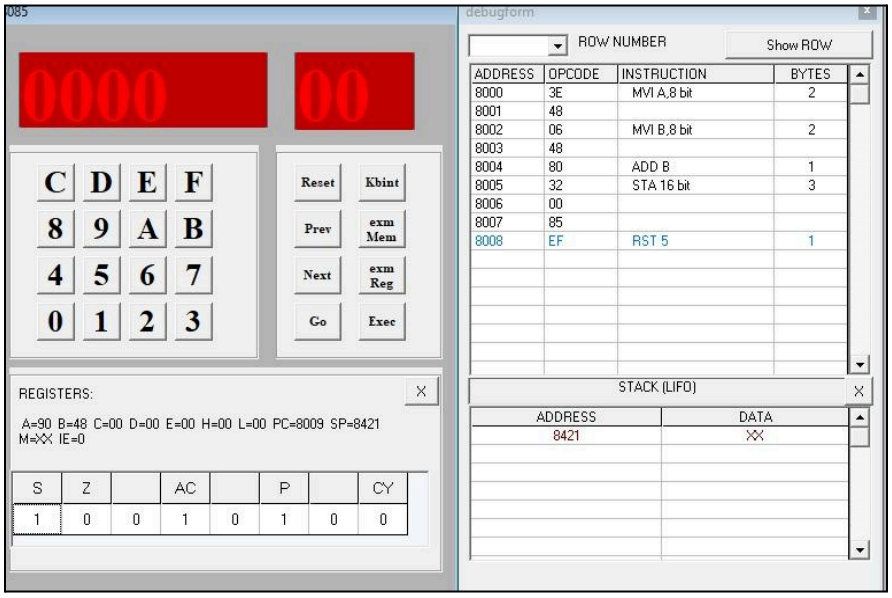

Fig 2.13

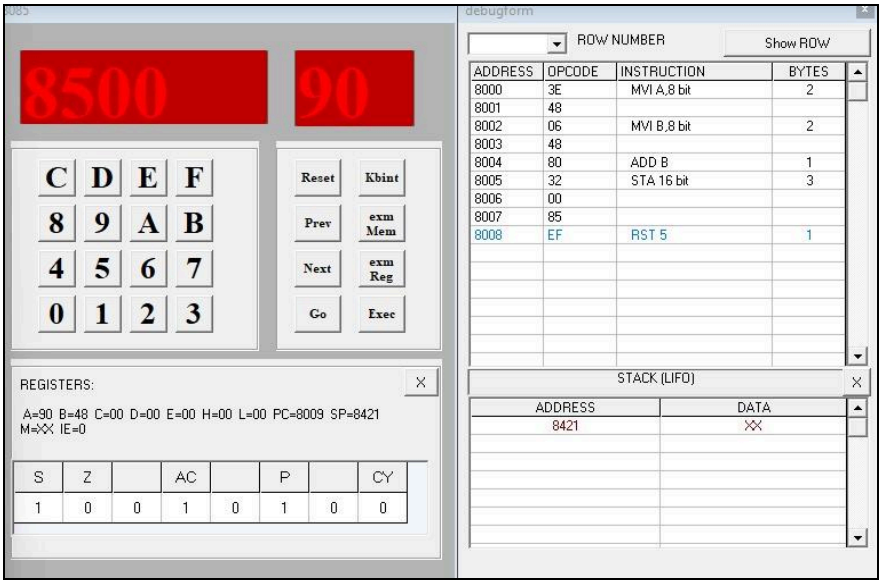
8500
90
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=90 B=48 C=00 D=00 E=00 H=00 L=00 PC=8009 SP=8421
M=XX IE=0
S Z AC P CY
1 0 0 1 0 1 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3E MVI A,8 bit 2
8001 48
8002 06 MVI B,8 bit 2
8003 48
8004 80 ADD B 1
8005 32 STA 16 bit 3
8006 00
8007 85
8008 EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 2.14

Program 2.3

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

Direct Addressing Mode:-

CODE	MEMORY LOCATION	OPCODE
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 2.3

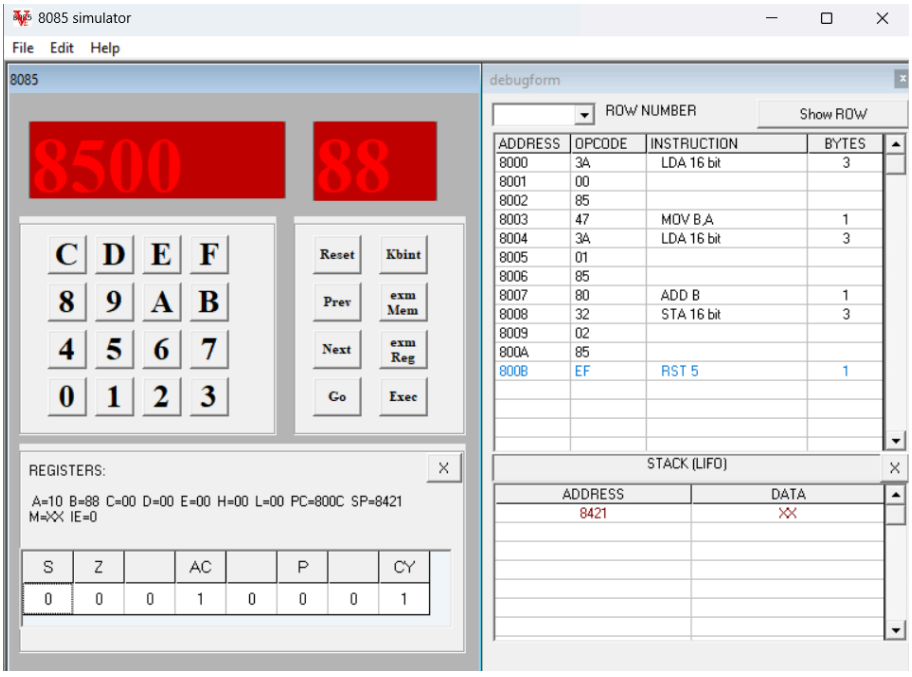

Fig 2.15

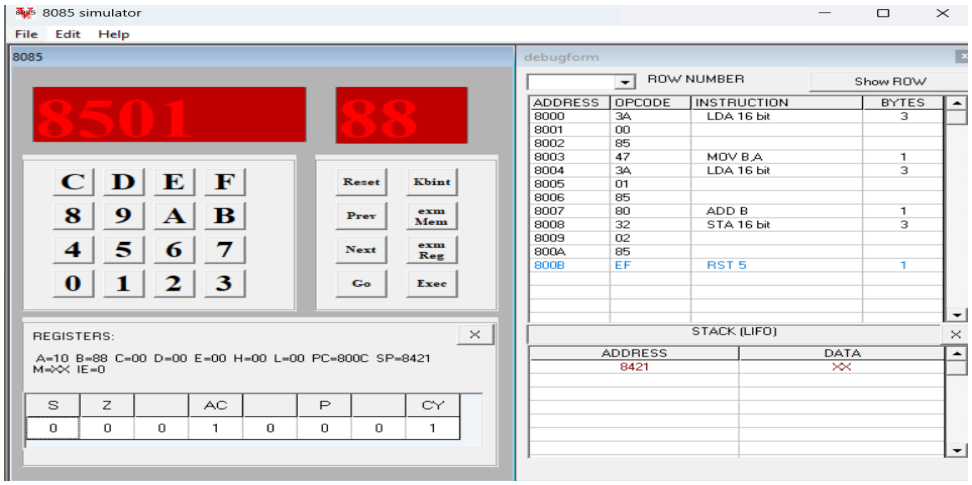

Fig 2.16

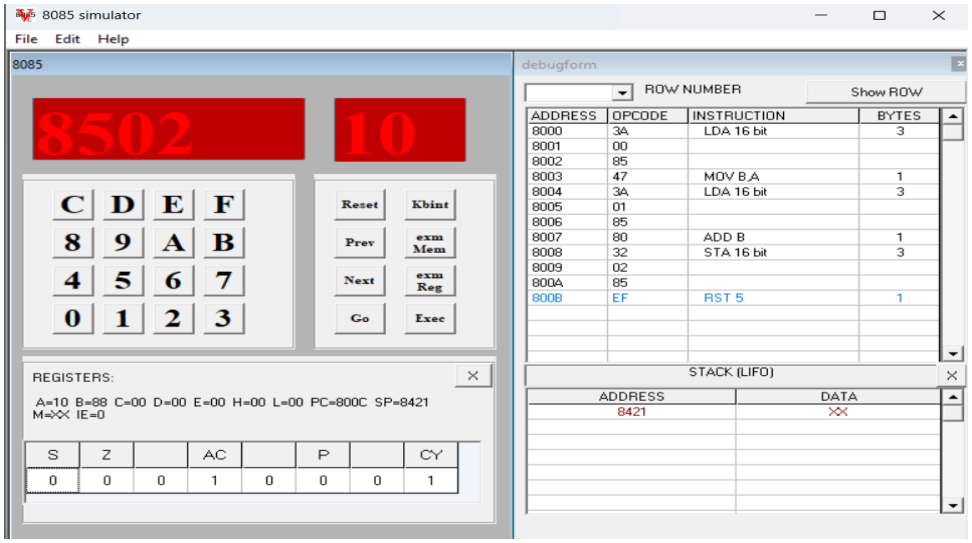

Fig 2.17

Indirect Addressing Mode:-

CODE	MEMORY LOCATION	OPCODE
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 2.4

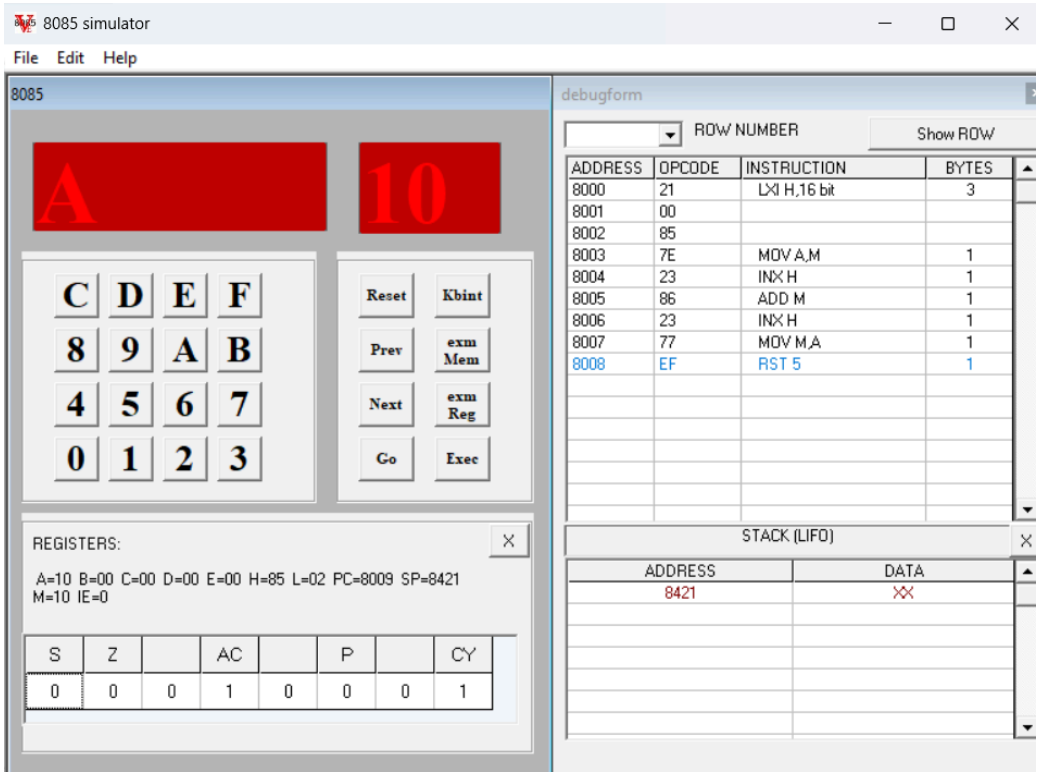

Fig 2.18

Program 2.4

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

Direct Addressing Mode:-

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 2.5

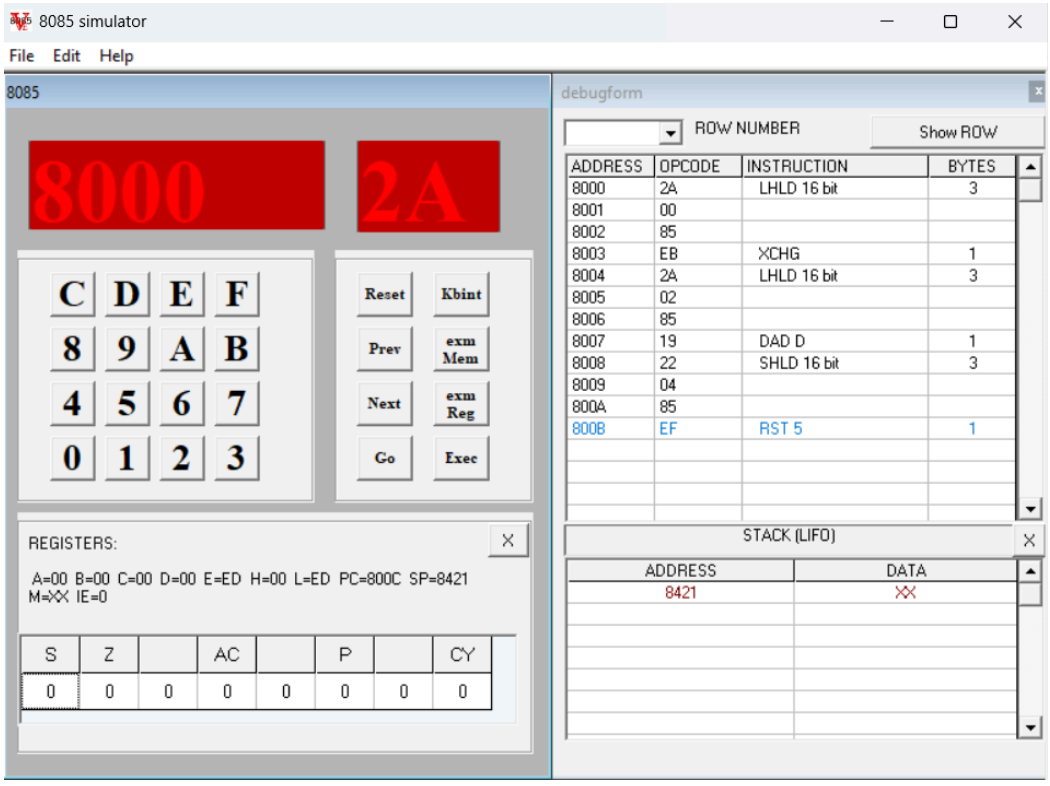

Fig 2.19

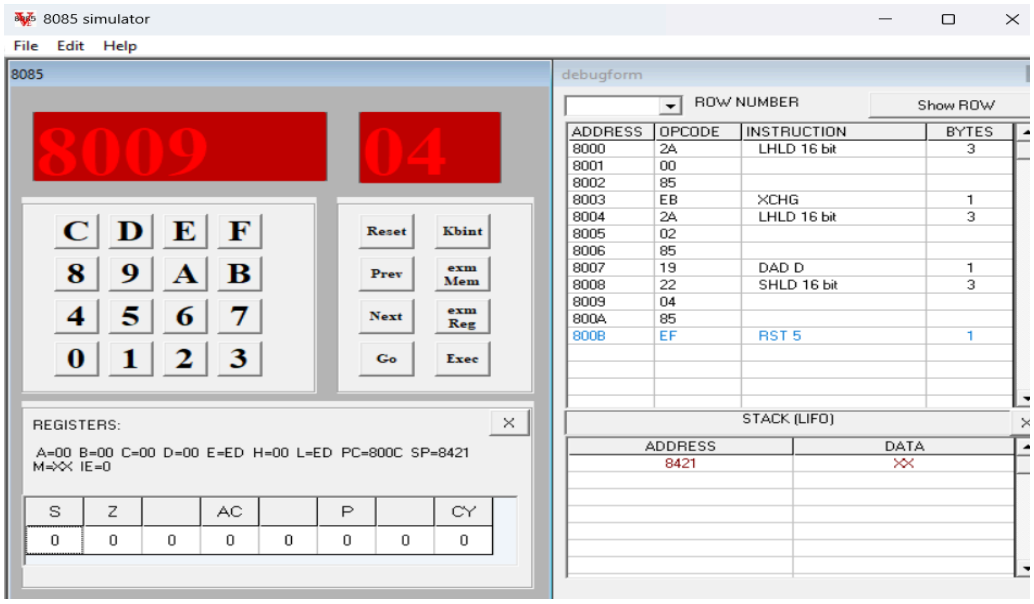

Fig 2.20

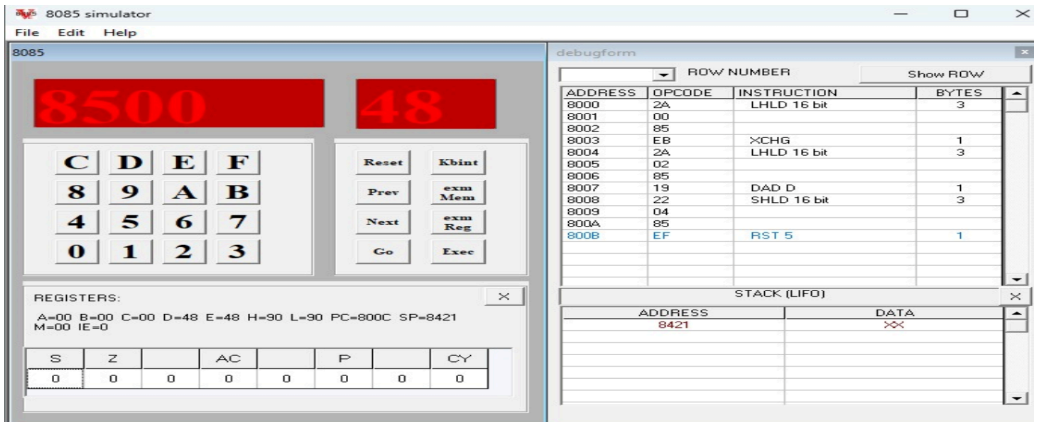

Fig 2.21

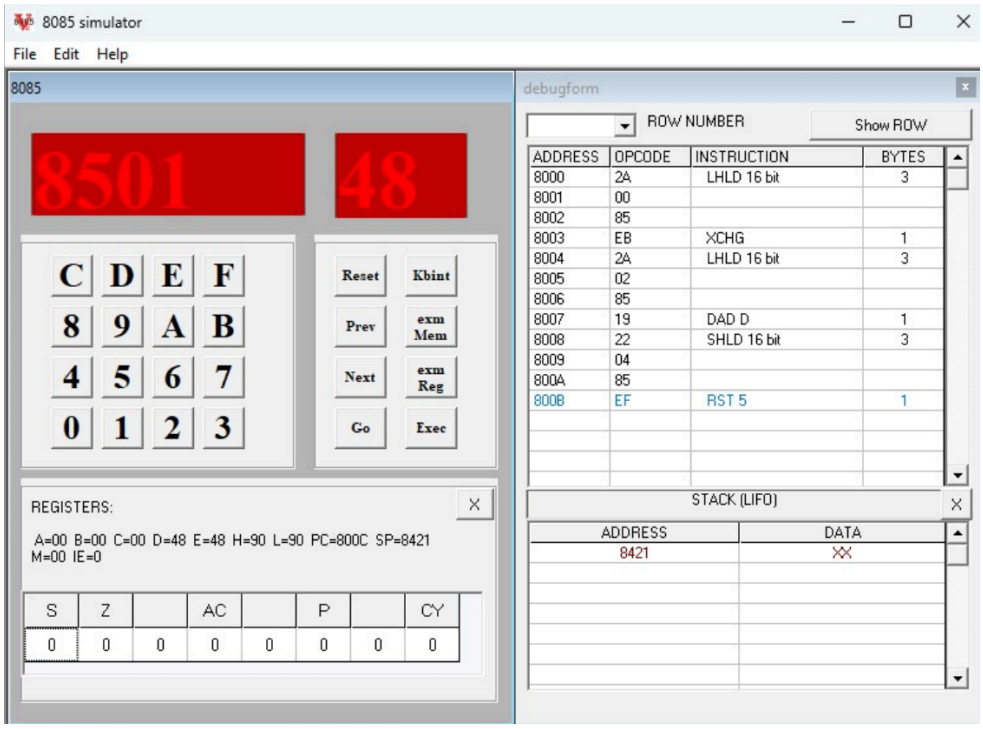

Fig 2.22

Indirect Addressing Mode:-

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 2.6

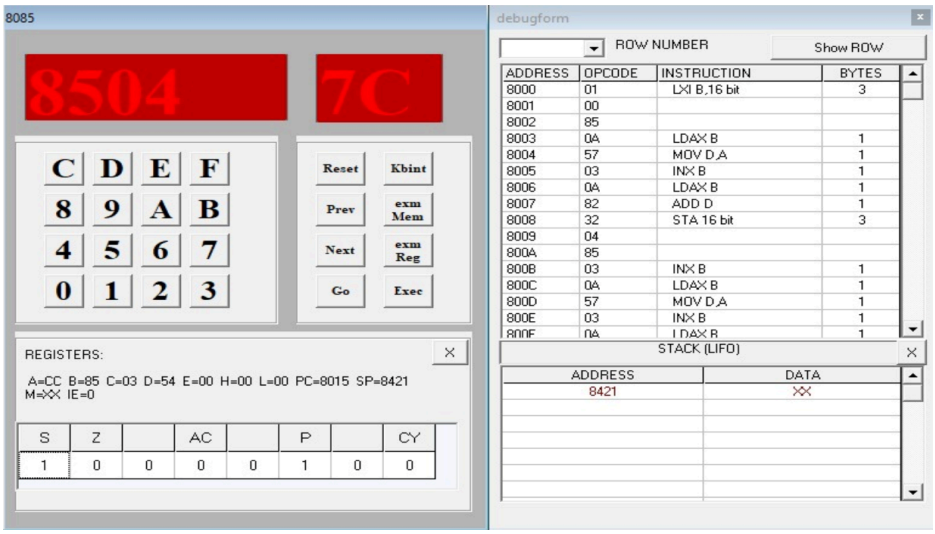

Fig 2.23

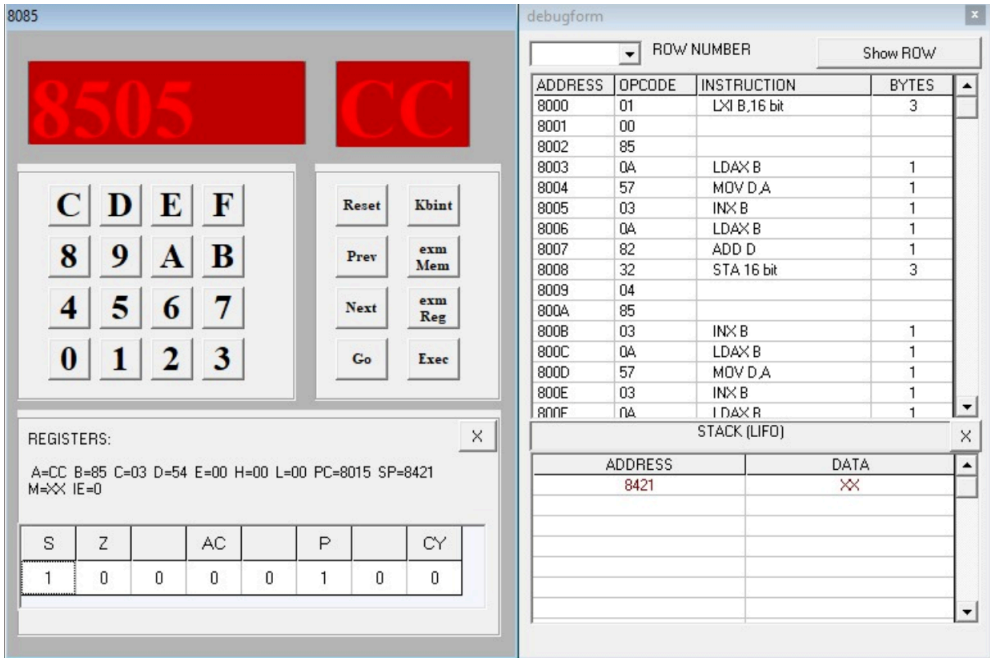
8085
8505
CC
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=CC B=85 C=03 D=54 E=00 H=00 L=00 PC=8015 SP=8421
M=XX IE=0
S Z AC P CY
1 0 0 0 0 1 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 01 LXI B,16 bit 3
8001 00
8002 85
8003 0A 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
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 2.24

Program 2.5

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

CODE	MEMORY LOCATION	OPCODE
MVI C, 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 2.7

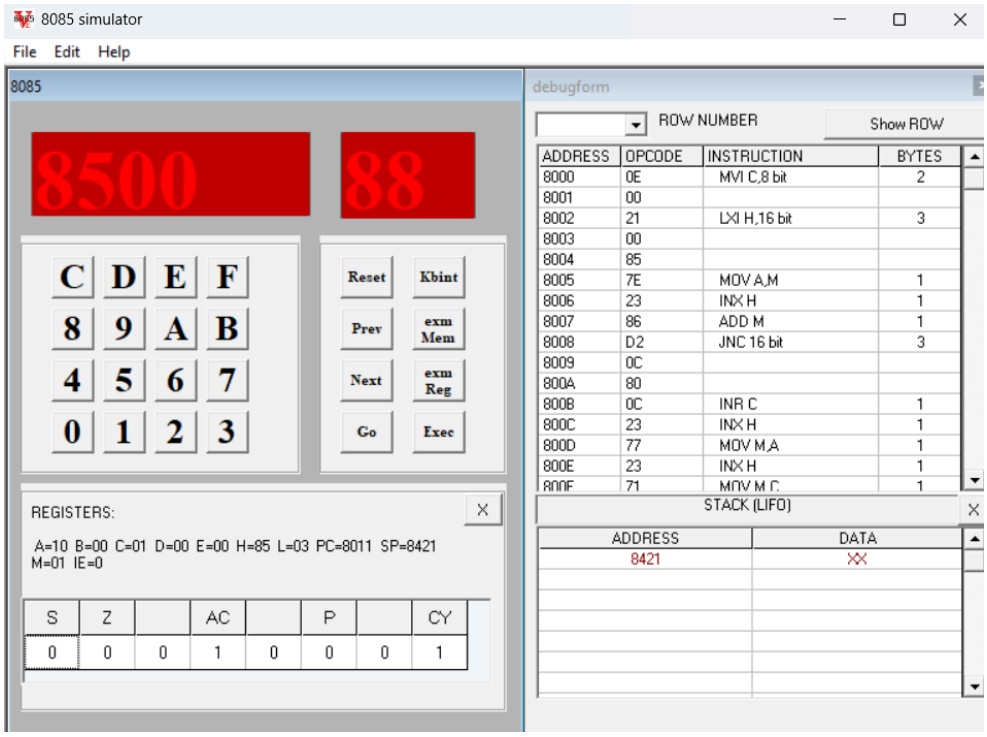

Fig 2.25

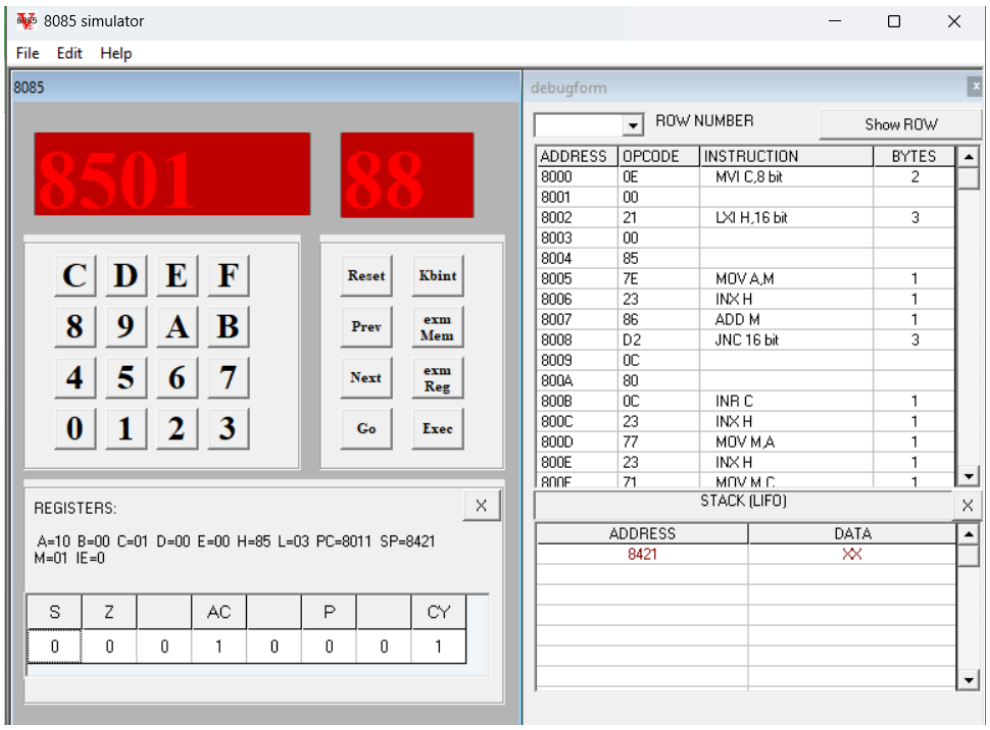

Fig 2.26

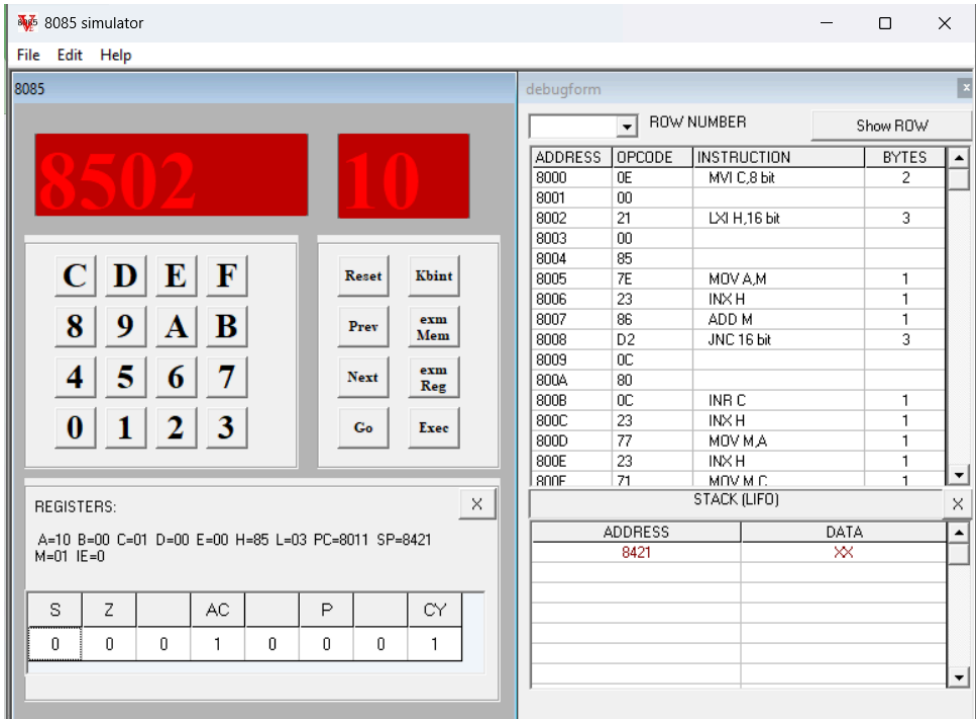

Fig 2.27

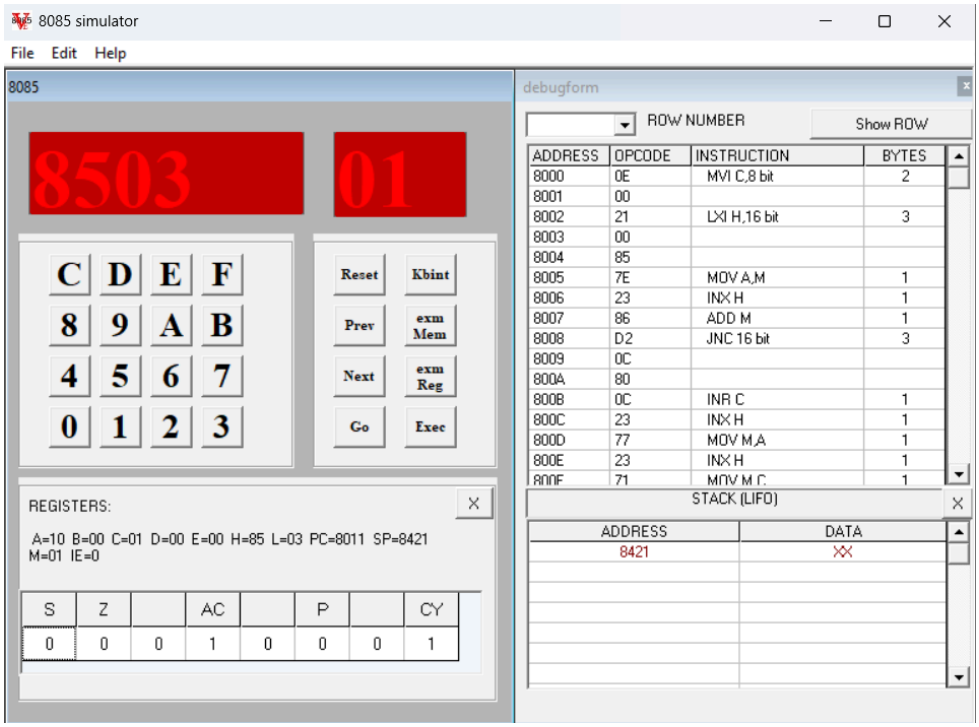

Fig 2.28

Program 2.6

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

1's Complement:-

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 2.8

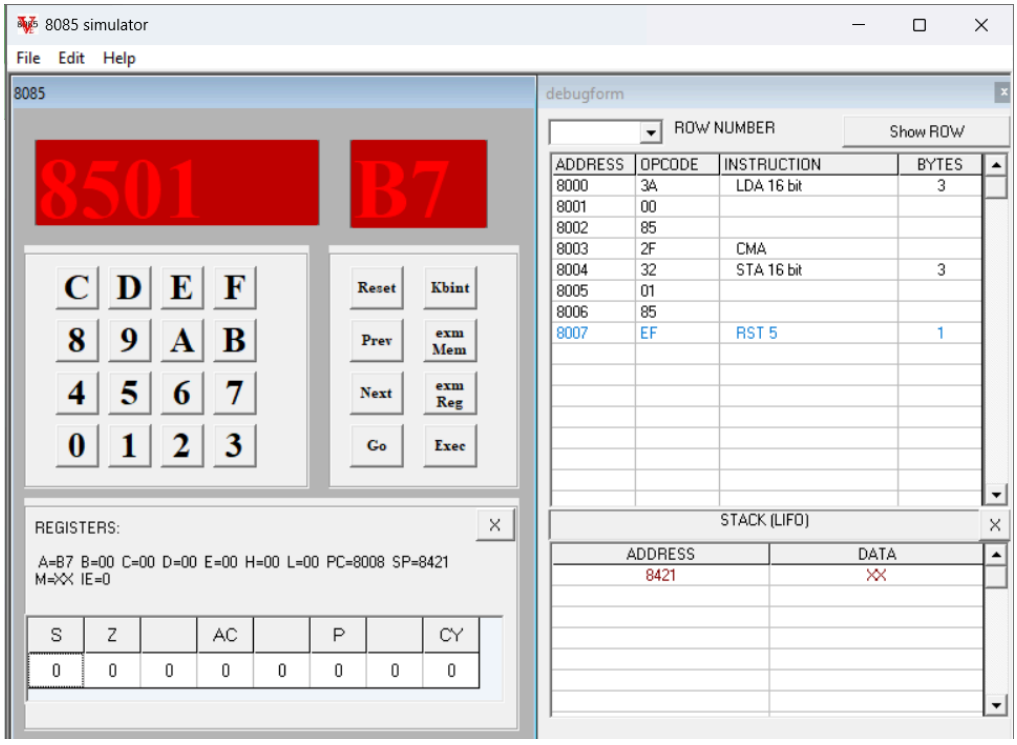

Fig 2.29

2’s Complement:-

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 2.9

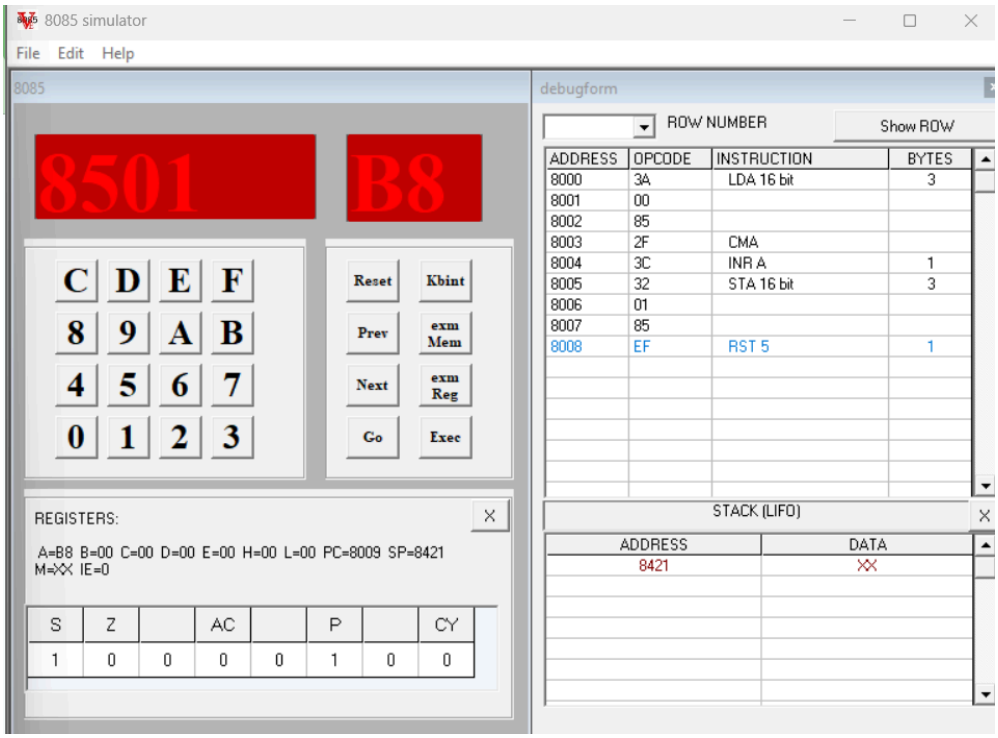

Fig 2.30

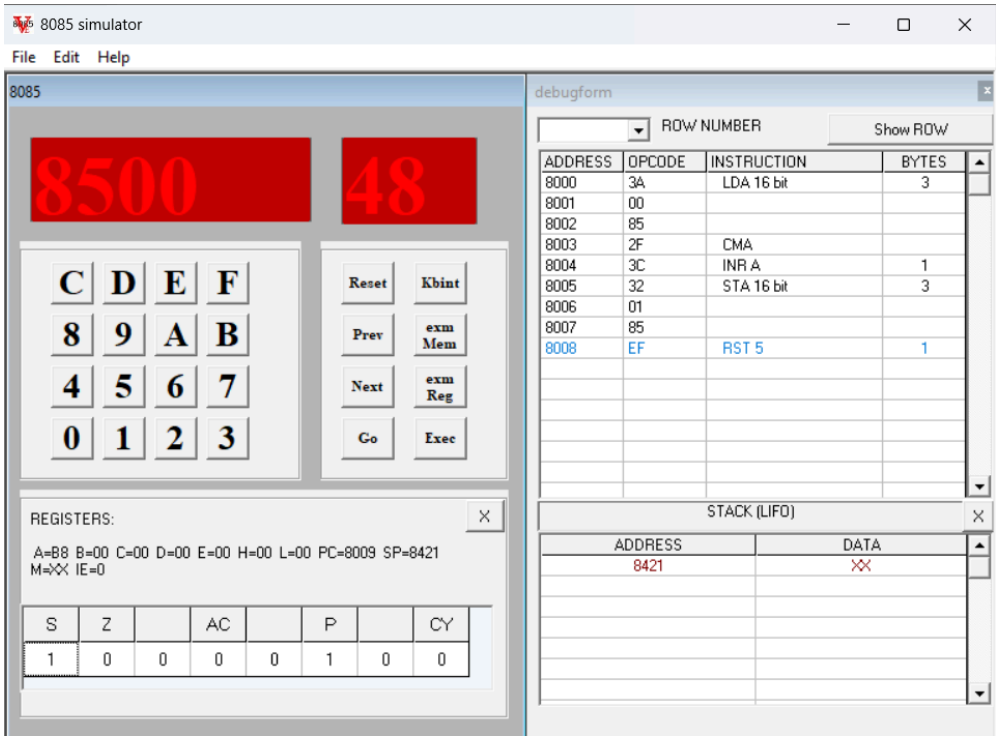
8085 simulator
File Edit Help
8085
8500
48
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=B8 B=00 C=00 D=00 E=00 H=00 L=00 PC=8009 SP=8421
M=XX IE=0
S Z AC P CY
1 0 0 0 0 1 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3A LDA 16 bit 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
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 2.31

Experiment No.-3

Aim: Write a program for the sum of a 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

Input - [8500] – 04, [8501] – 9A, [8502] – 52, [8503] – 89, [8504] – 3E
Output - [8600] – B3

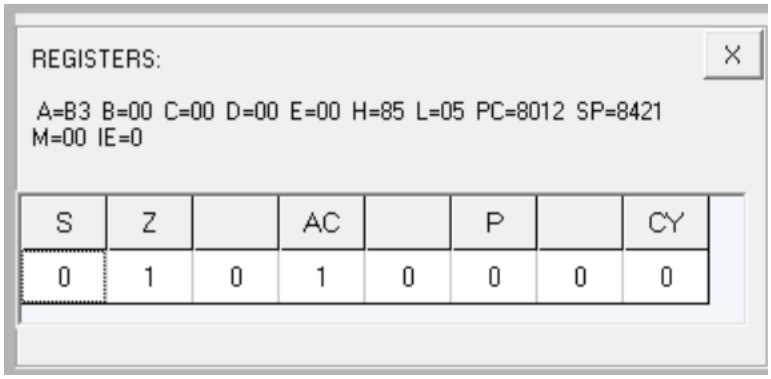

Fig 3.1

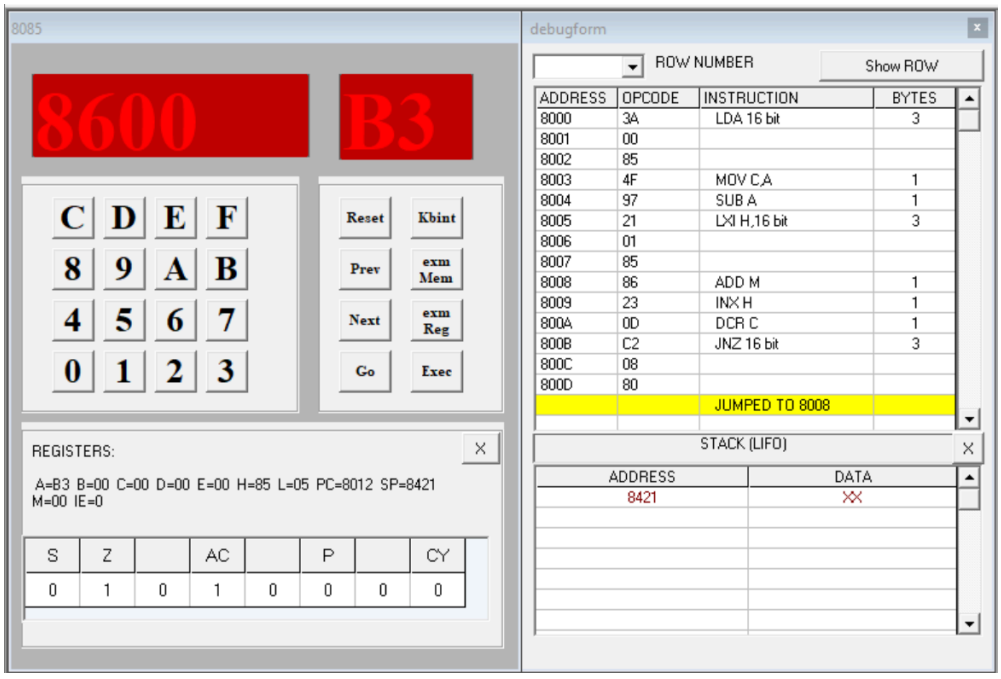
8085
8600
B3
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=B3 B=00 C=00 D=00 E=00 H=85 L=05 PC=8012 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 1 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3A LDA 16 bit 3
8001 00
8002 85
8003 4F MOV 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
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 3.2

Experiment No.-4

Aim: 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	21,00,86
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

Input

[8500] – 01, [8501] – 02, [8502] – 03, [8503] – 04, [8504] – 05, [8505] – 06, [8506] – 07, [8507] – 08, [8508] – 09, [8509] – 0A

Output

[8600] – 01, [8601] – 02, [8602] – 03, [8603] – 04, [8604] – 05, [8605] – 06, [8606] – 07, [8607] – 08, [8608] – 09, [8609] – 0A

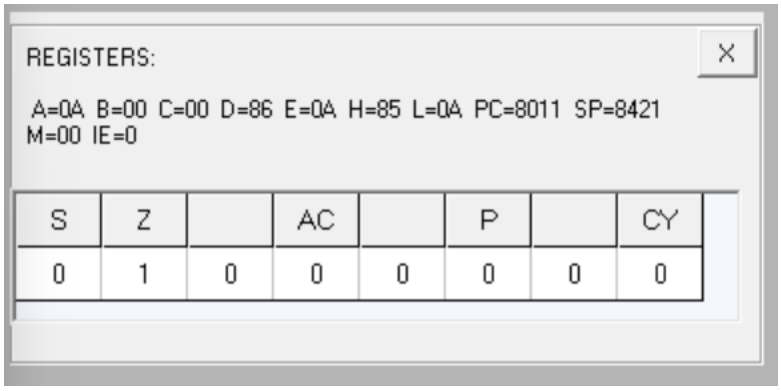
REGISTERS:
A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0

Fig 4.1

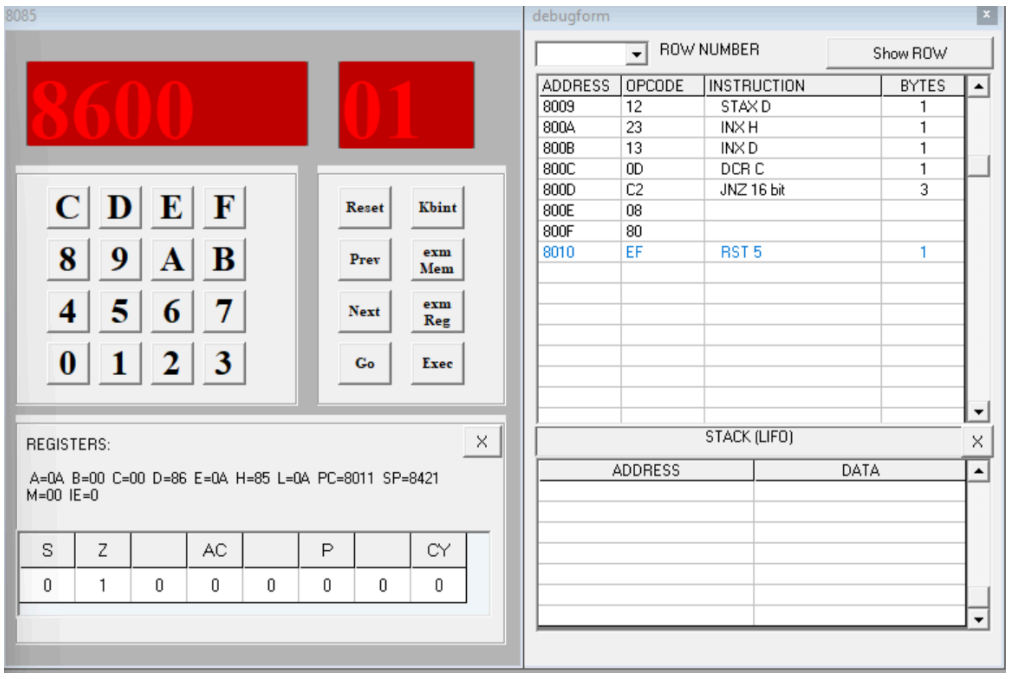
8085
8600
01
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
8010 EF RST 5 1
STACK (LIFO)
ADDRESS DATA

Fig 4.2

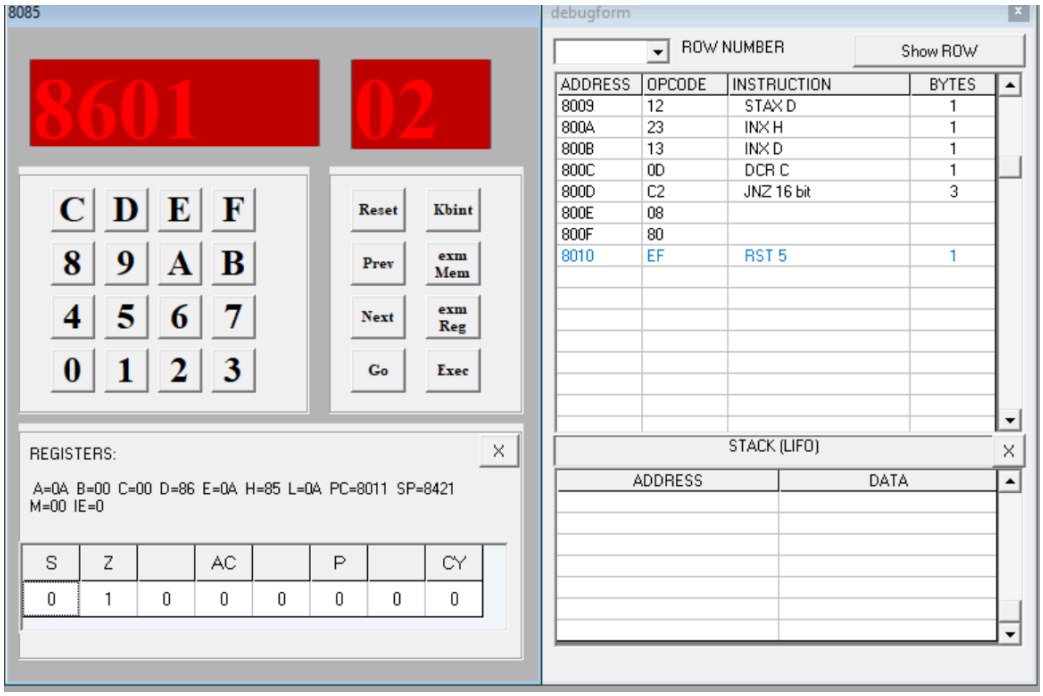
8085
8601
02
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
8010 EF RST 5 1
STACK (LIFO)
ADDRESS DATA

Fig 4.3

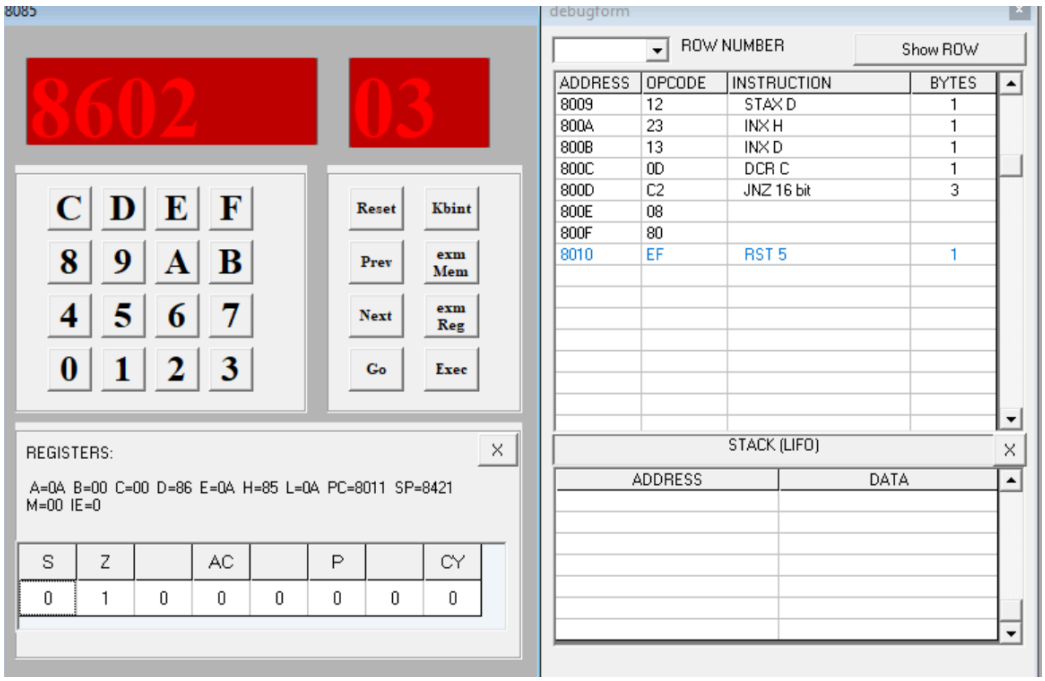
8085
8602
03
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
8010 EF RST 5 1
STACK (LIFO)
ADDRESS DATA

Fig 4.4

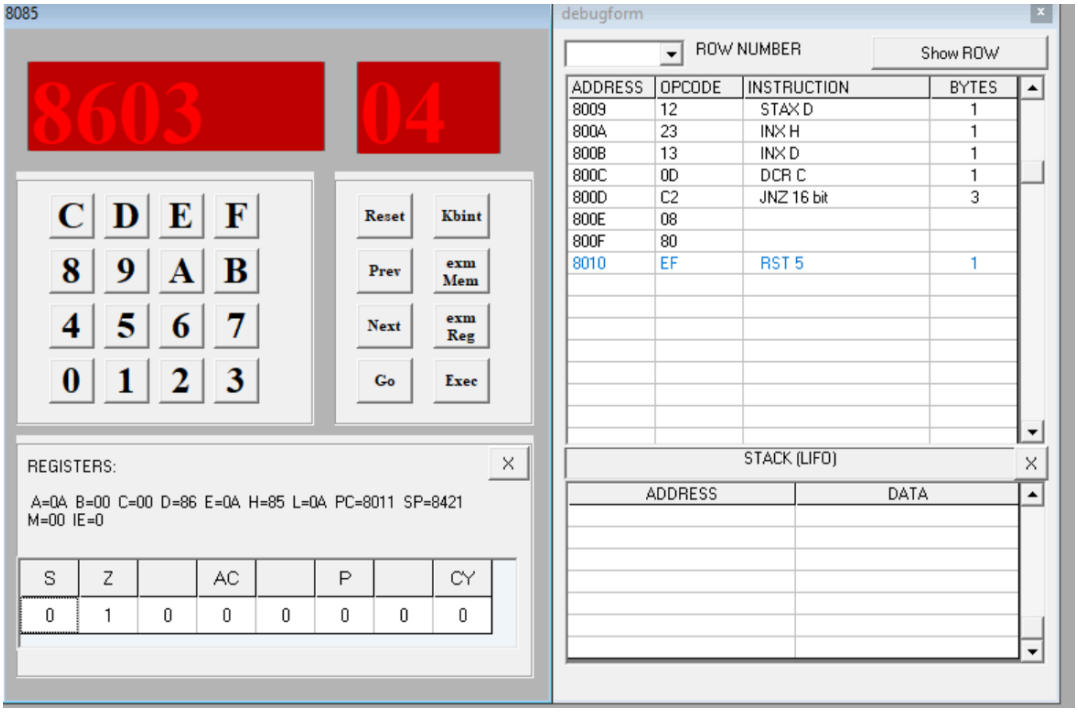
8085
8603
04
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
8010 EF RST 5 1
STACK (LIFO)
ADDRESS DATA

Fig 4.5

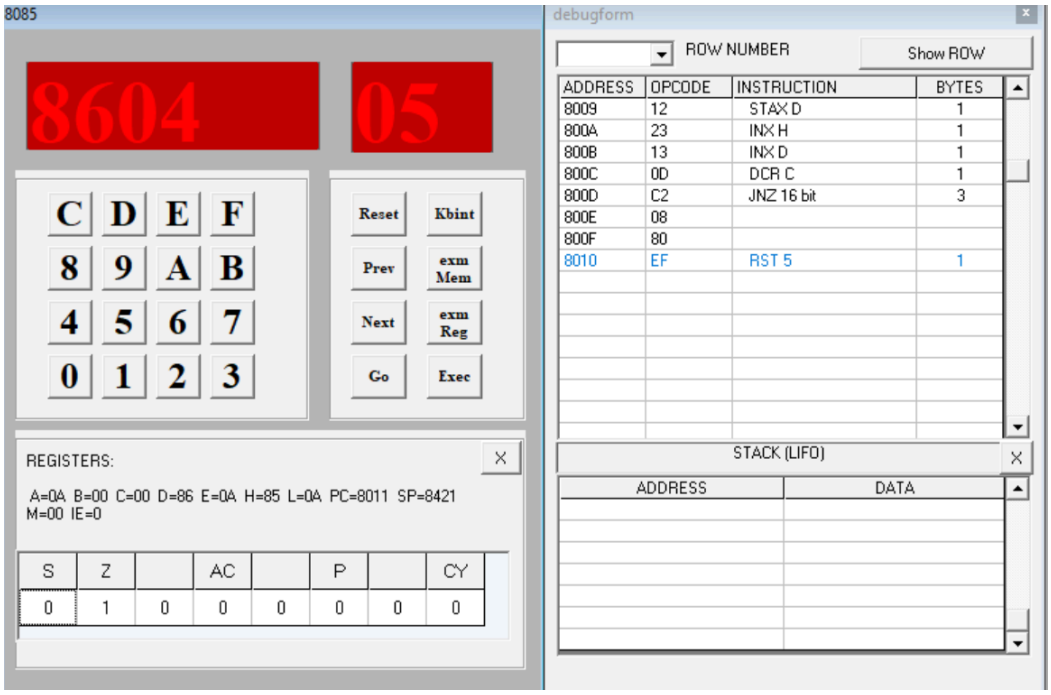
8085
8604
05
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
8010 EF RST 5 1
STACK (LIFO)
ADDRESS DATA

Fig 4.6

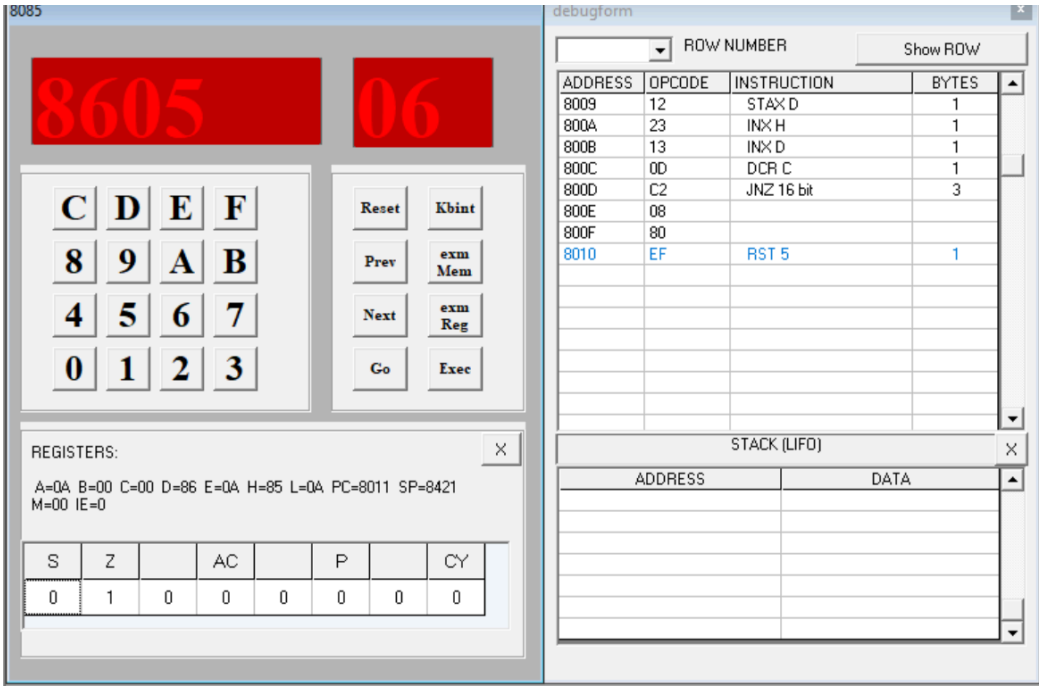
8085
8605
06
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
8010 EF RST 5 1
STACK (LIFO)
ADDRESS DATA

Fig 4.7

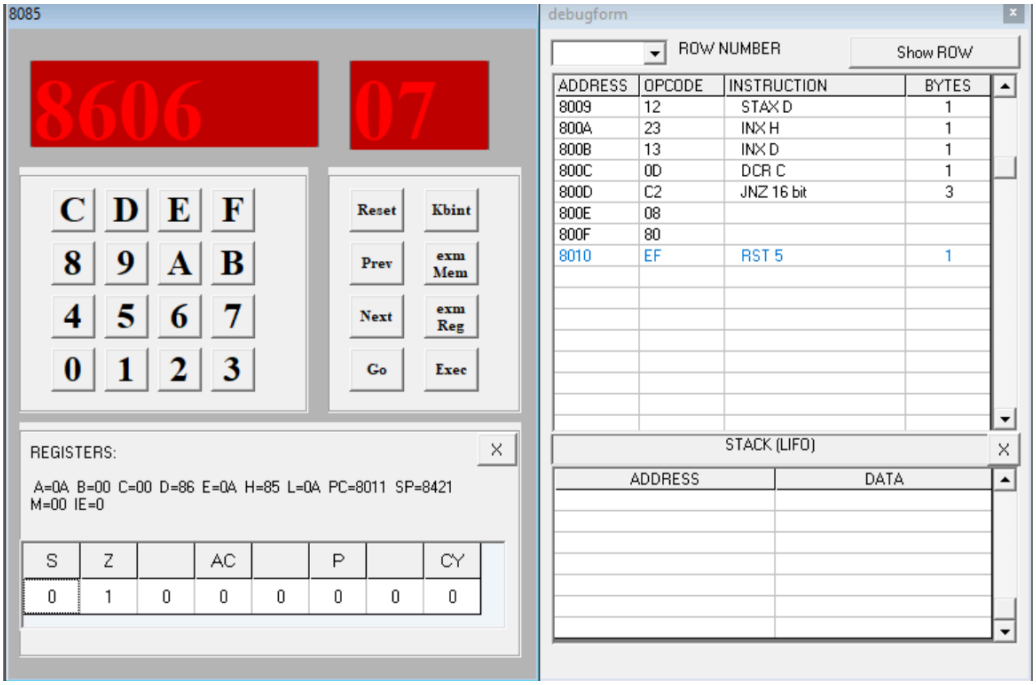
8085
8606
07
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
8010 EF RST 5 1
STACK (LIFO)
ADDRESS DATA

Fig 4.8

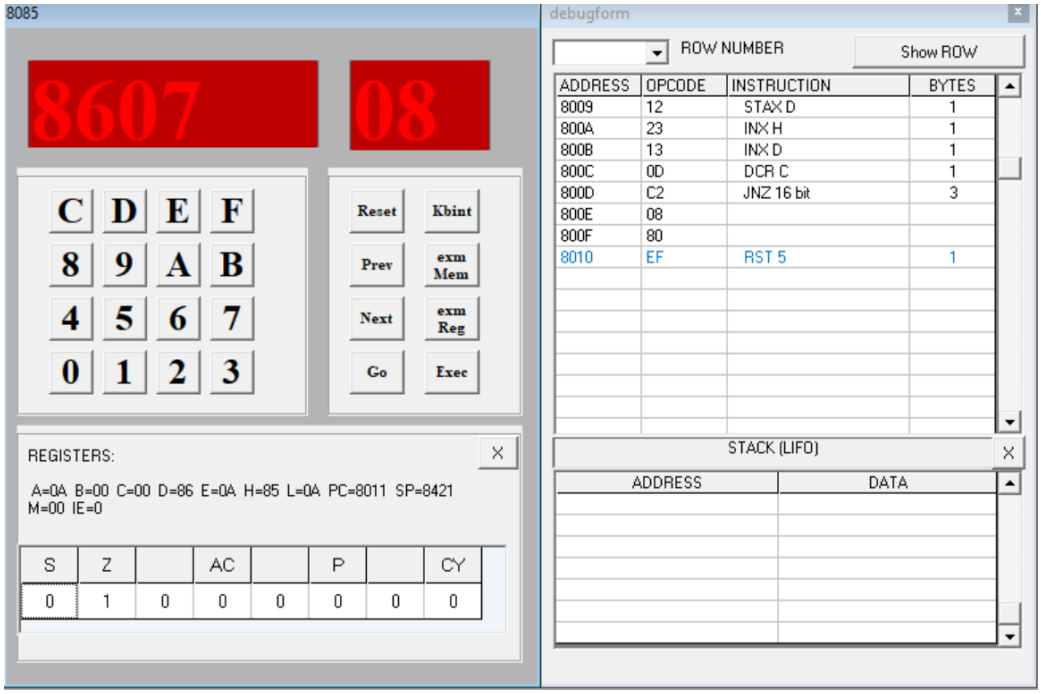

Fig 4.9

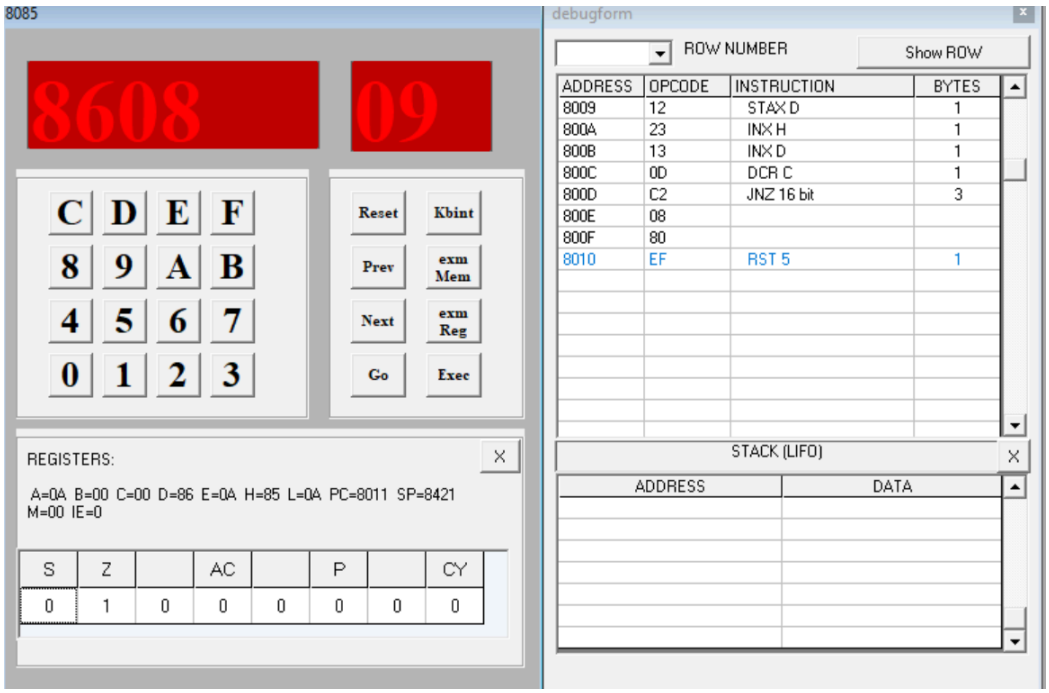

Fig 4.10

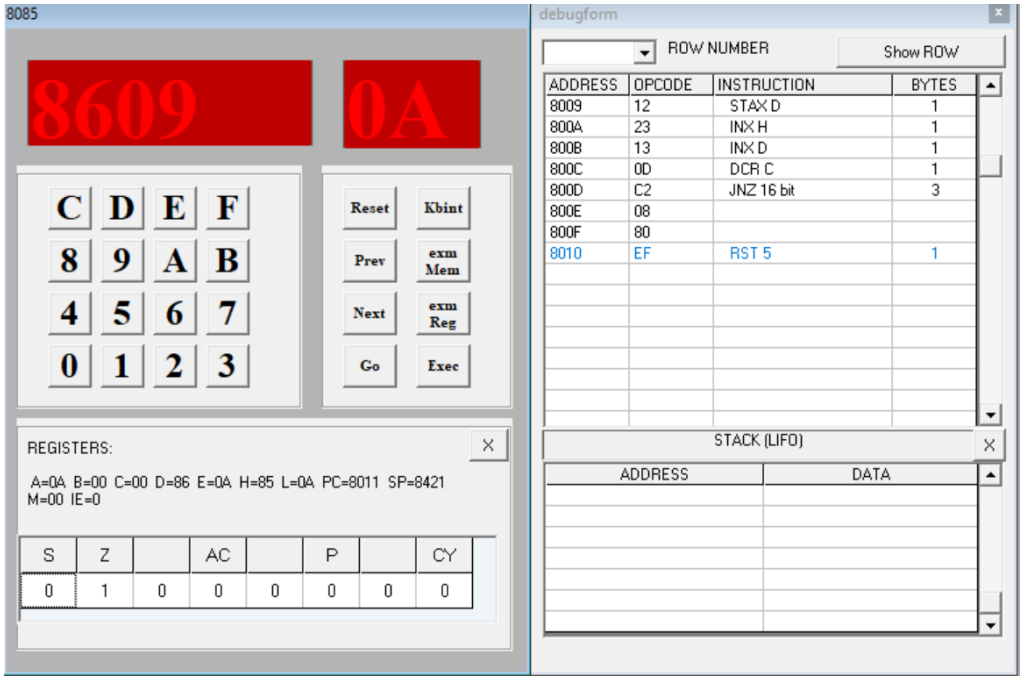
8085
8609
0A
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
8010 EF RST 5 1
STACK (LIFO)
ADDRESS DATA

Fig 4.11

Experiment No.-5

Aim: 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	22,00,86

Table 5

Input - [8500] – B2, [8501] – 03
Result – B2 + B2 + B2 = 0216 H Output - [8600] – 16, [8601] – 02

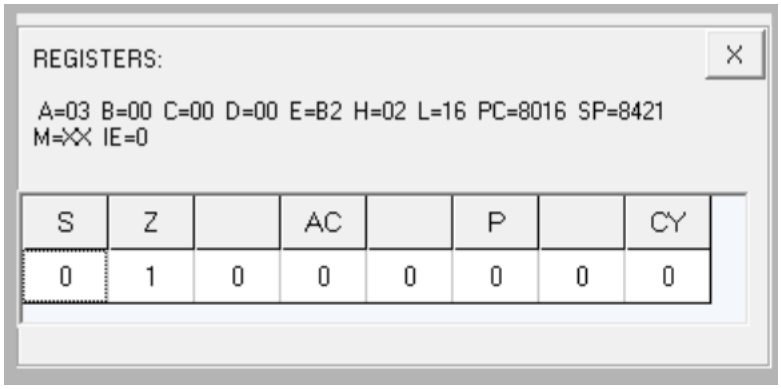

Fig 5.1

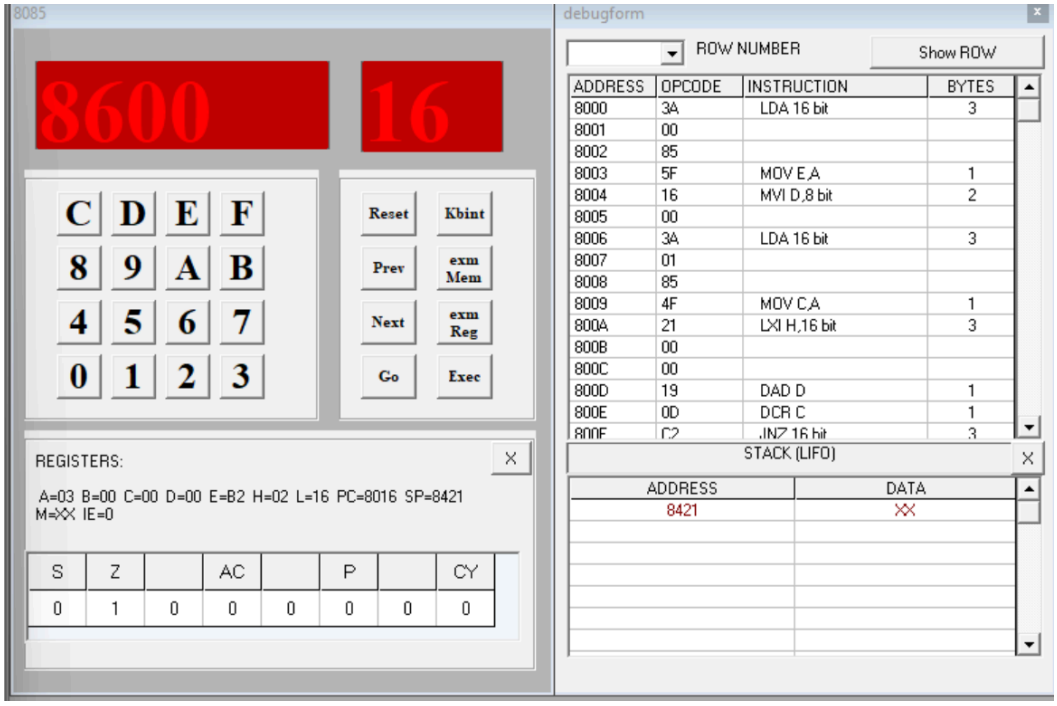
8085
8600
16
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=03 B=00 C=00 D=00 E=B2 H=02 L=16 PC=8016 SP=8421
M=XX IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3A LDA 16 bit 3
8001 00
8002 85
8003 5F MOV 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)
ADDRESS DATA
8421 XX

Fig 5.2

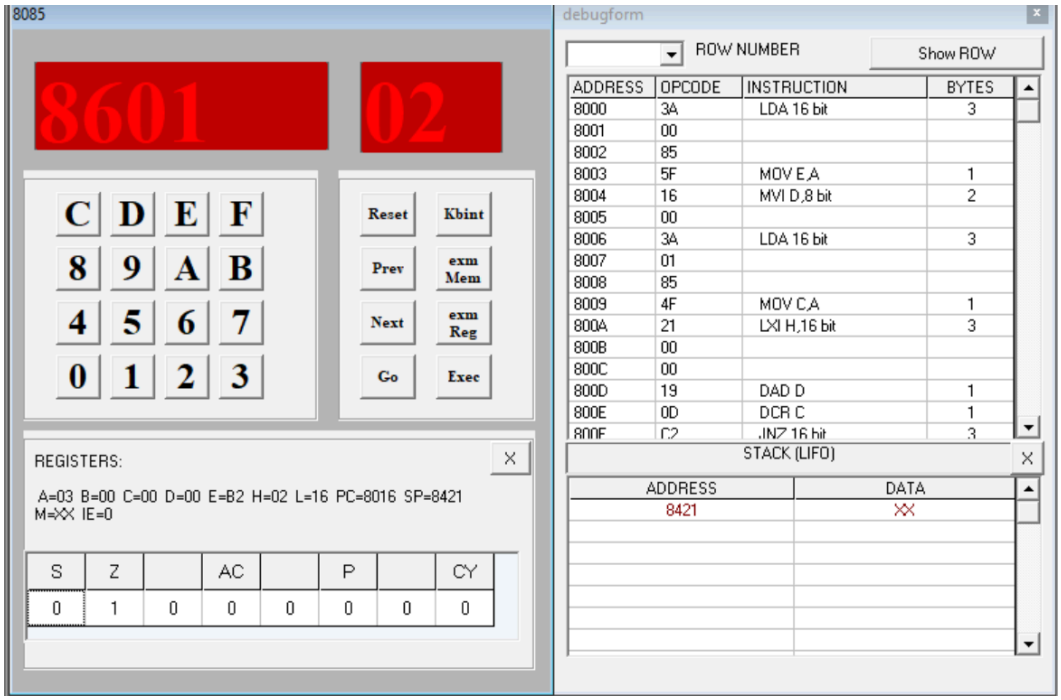
8085
8601
02
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=03 B=00 C=00 D=00 E=B2 H=02 L=16 PC=8016 SP=8421
M=XX IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3A LDA 16 bit 3
8001 00
8002 85
8003 5F MOV 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)
ADDRESS DATA
8421 XX

Fig 5.3

Experiment No.-6

Aim: 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 addresses.

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

Table 6

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

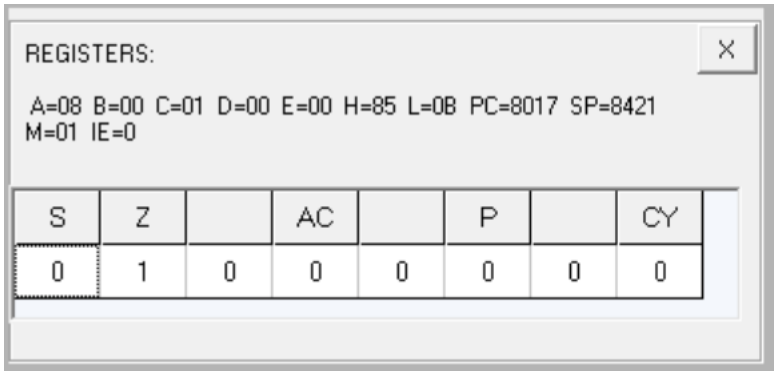

Fig 6.1

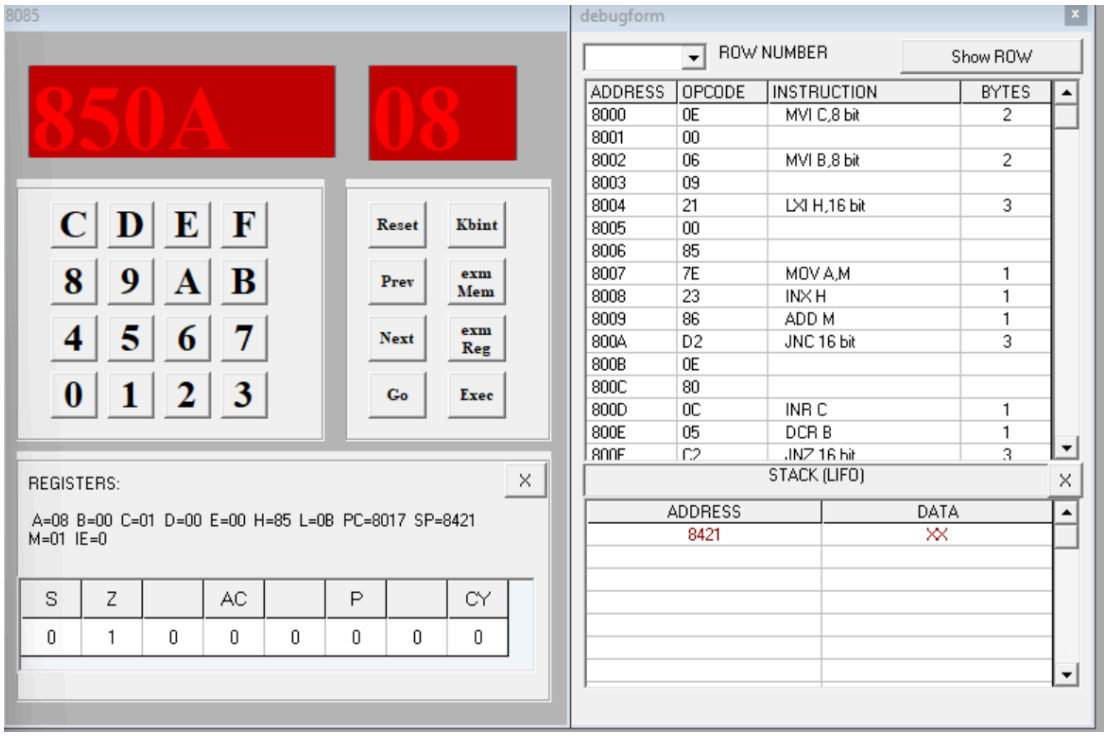

Fig 6.2

Experiment No.-7

Aim: 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

Input

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

Output

[8600] – 02

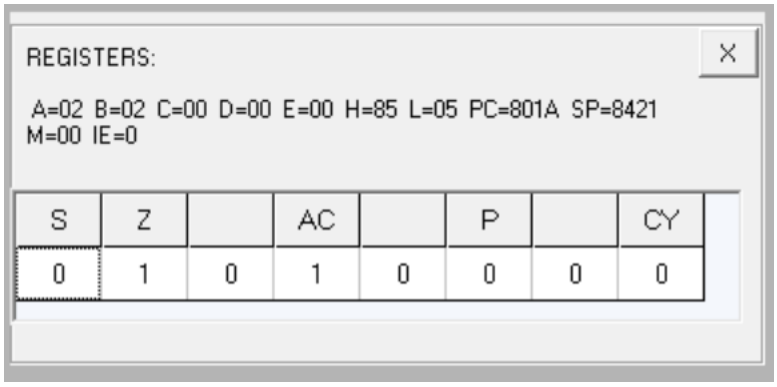
REGISTERS:
A=02 B=02 C=00 D=00 E=00 H=85 L=05 PC=801A SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 1 0 0 0 0

Fig 7.1

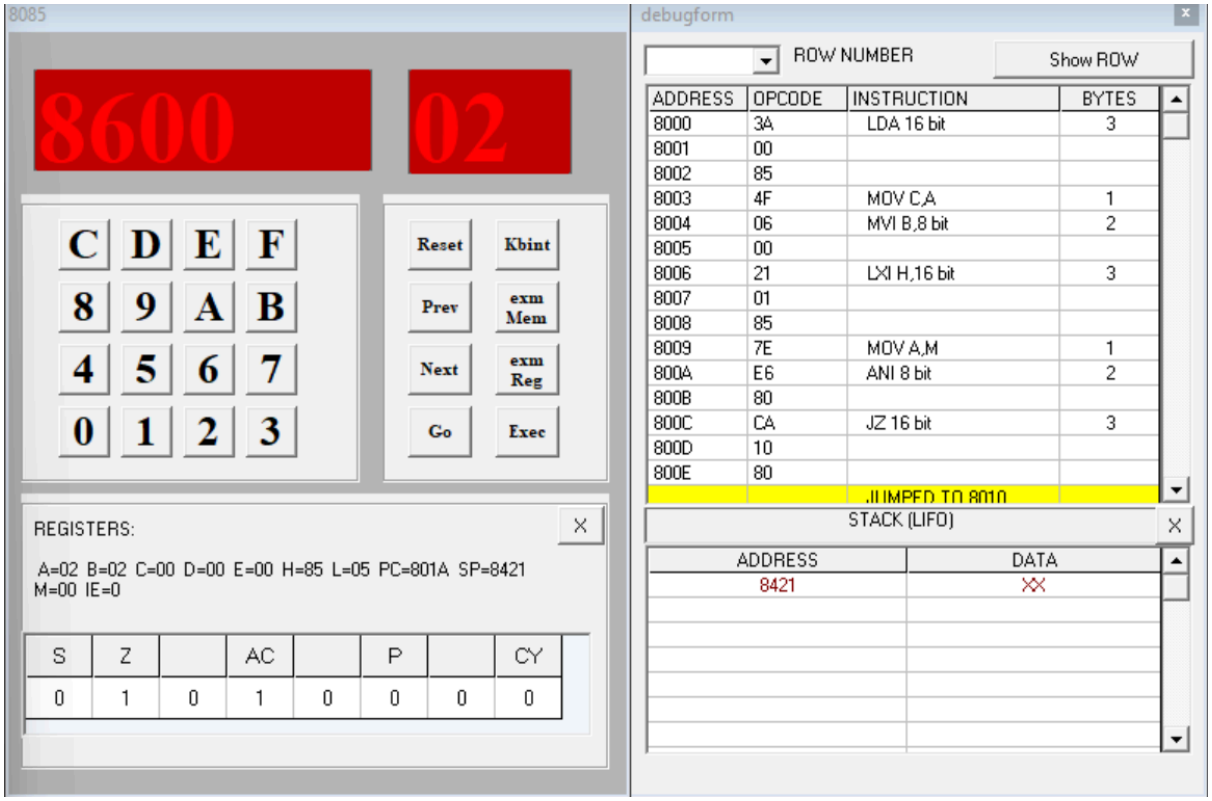
8085
8600
02
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=02 B=02 C=00 D=00 E=00 H=85 L=05 PC=801A SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 1 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3A LDA 16 bit 3
8001 00
8002 85
8003 4F MOV 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
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 7.2

Experiment No.-8

Aim: 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	D2,10,80
LDA 8500H	800B	3A,00,85
MVI B, 08	800C, 800D	06,08
MVI D,00	800E,800F	16,00
Loop: RLC	8010	07
JNC Loop2	8011,8012, 8013	D2,10,80
LDA 8500H	8014, 8015, 8016	3A,00,85
MVI B, 08	8017,8018	06,08
MVI D, 00	8019, 801A	16,00

Table 8

Input - [8500] – 25 0010 0101

Output - [8600] – 03


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

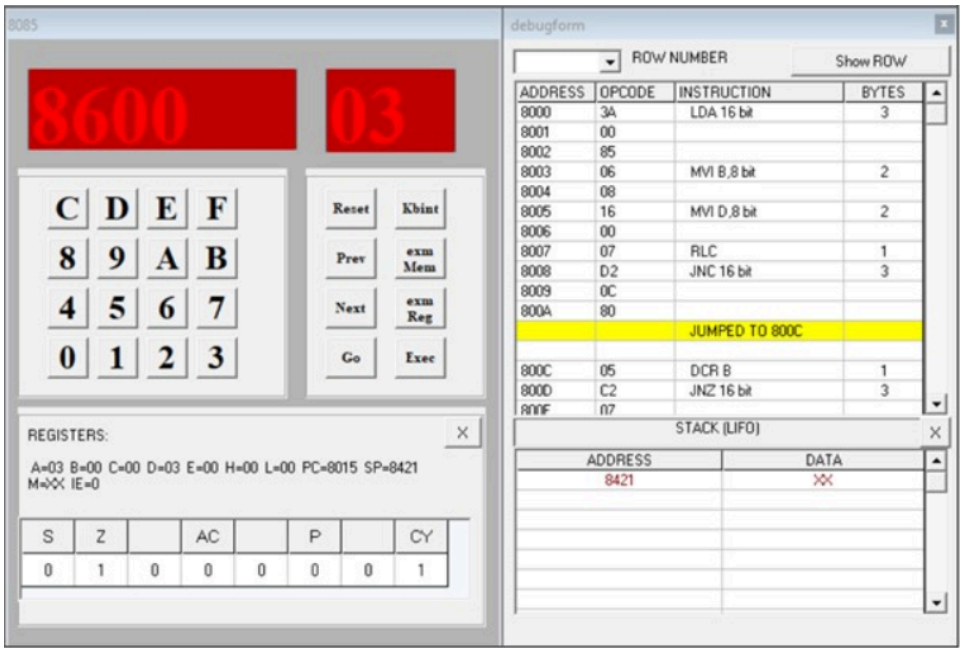
8600
03
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=03 B=00 C=00 D=03 E=00 H=00 L=00 PC=8015 SP=8421
M=XX IE=0
S Z AC P CY
0 1 0 0 0 0 0 1
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3A LDA 16 bit 3
8001 00
8002 85
8003 06 MVI B,8 bit 2
8004 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
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 8.2

Experiment No.-9

Aim: Write a program to arrange numbers in Ascending order.

CODE	MEMORY LOCATION	OPCODE
LXI H, 8500H	8000, 8001, 8002	21,00,85
MOV C,M	8003	4E
DCR C	8004	0D
Repeat: MOV D,C	8005	51
LXI H, 8501H	8006,8007,8008	21,01,85
Loop: MOV A,M	8009	7E
INX H	800A	23
CMP M	800B	BD
JC Skip	800C,800D,800E	DA,14,80
MOV B,M	800F	46
MOV M, A	8010	77
DCX H	8011	2B
MOV M,B	8012	70
INX H	8013	23
Skip: DCR D	8014	15
JNZ Loop	8015,8016.8017	C2,09,80
DCR C	8018	0D
JNZ Repeat	8019,801A,801B	C2,05,80

Table 9

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

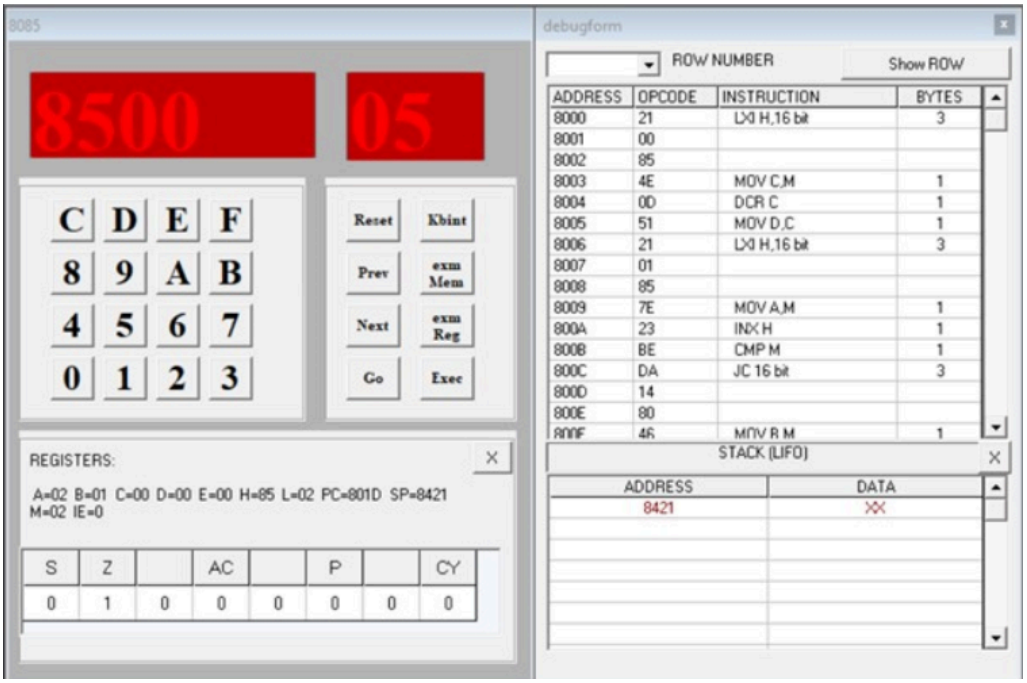
8085
8500
05
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=02 B=01 C=00 D=00 E=00 H=85 L=02 PC=801D SP=8421
M=02 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 21 LXI H,16 bit 3
8001 00
8002 85
8003 4E 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 B,M 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 9.1

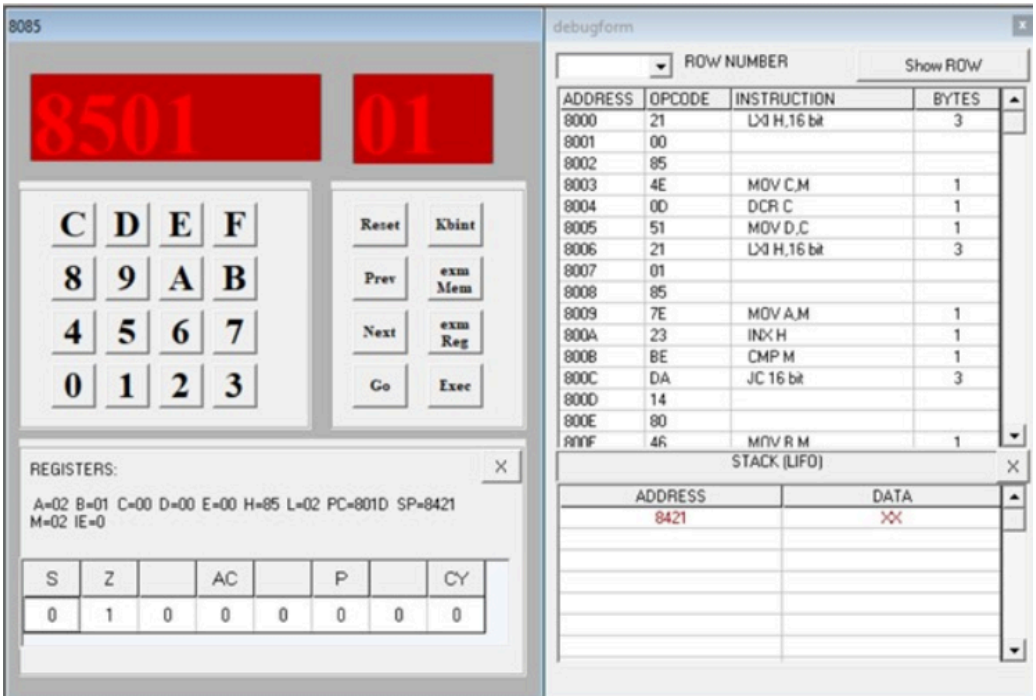
8085
8501
01
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=02 B=01 C=00 D=00 E=00 H=85 L=02 PC=801D SP=8421
M=02 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 21 LXI H,16 bit 3
8001 00
8002 85
8003 4E 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 B,M 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 9.2

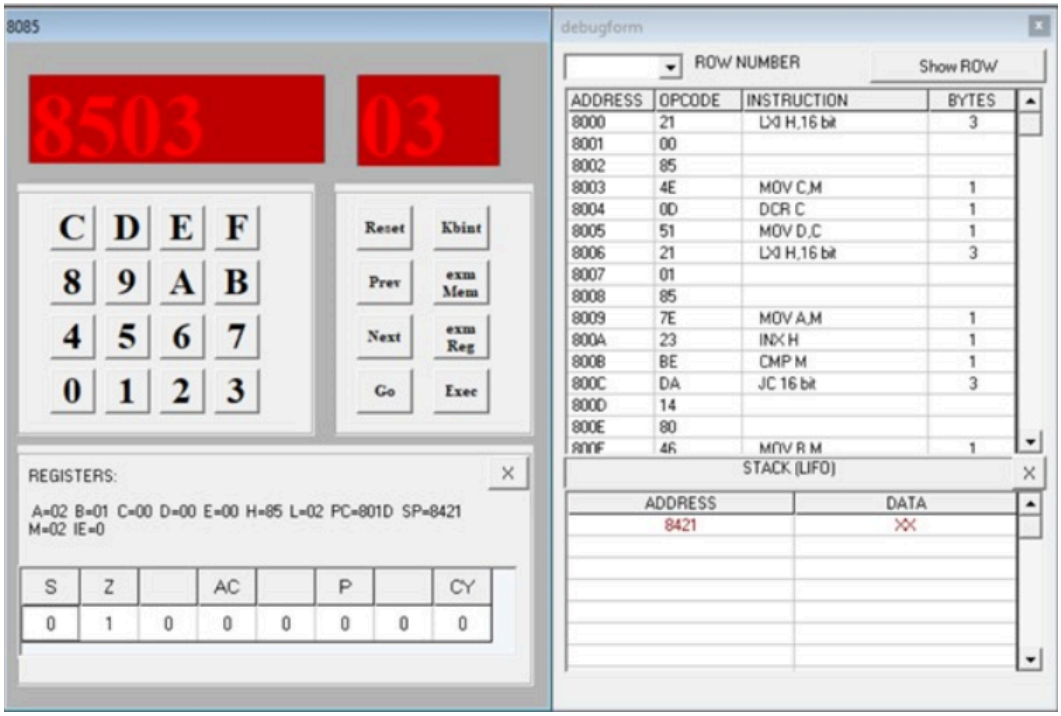
8085
8503
03
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=02 B=01 C=00 D=00 E=00 H=85 L=02 PC=801D SP=8421
M=02 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 21 LXI H,16 bit 3
8001 00
8002 85
8003 4E 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 B,M 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 9.3

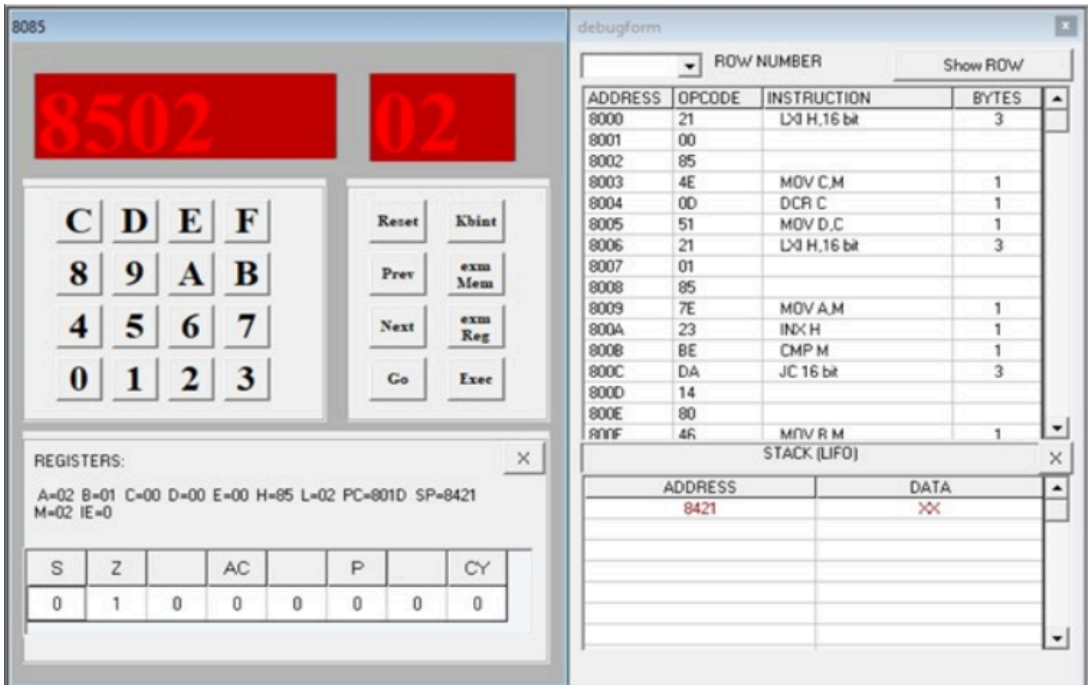
8085
8502
02
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=02 B=01 C=00 D=00 E=00 H=85 L=02 PC=801D SP=8421
M=02 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 21 LXI H,16 bit 3
8001 00
8002 85
8003 4E 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 B,M 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 9.4

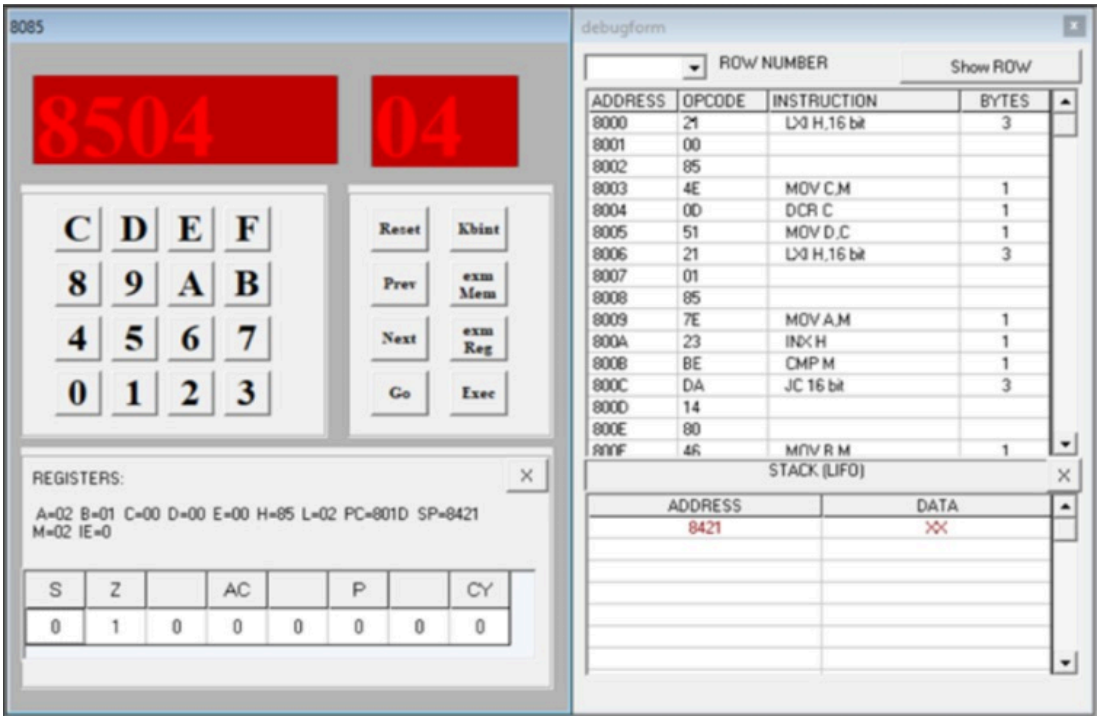
8085
8504
04
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=02 B=01 C=00 D=00 E=00 H=85 L=02 PC=801D SP=8421
M=02 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 21 LXI H,16 bit 3
8001 00
8002 85
8003 4E 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 B,M 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 9.5

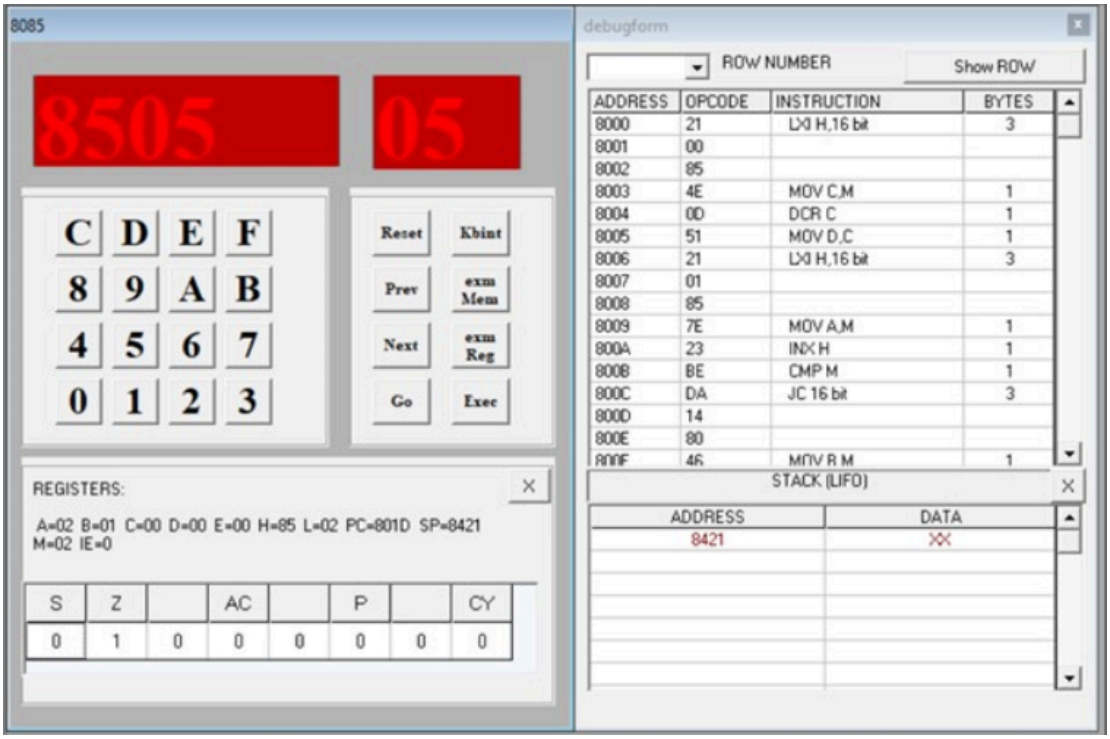
8085
8505
05
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=02 B=01 C=00 D=00 E=00 H=85 L=02 PC=801D SP=8421
M=02 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 21 LXI H,16 bit 3
8001 00
8002 85
8003 4E 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 B,M 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 9.6

Experiment No.-10

Aim: Write a program to calculate the sum of a 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

Input - [8500] – 04, [8501] – 20,

[8502] – 15 , [8503] – 13, [8504] – 22

Output - [8600] – 42

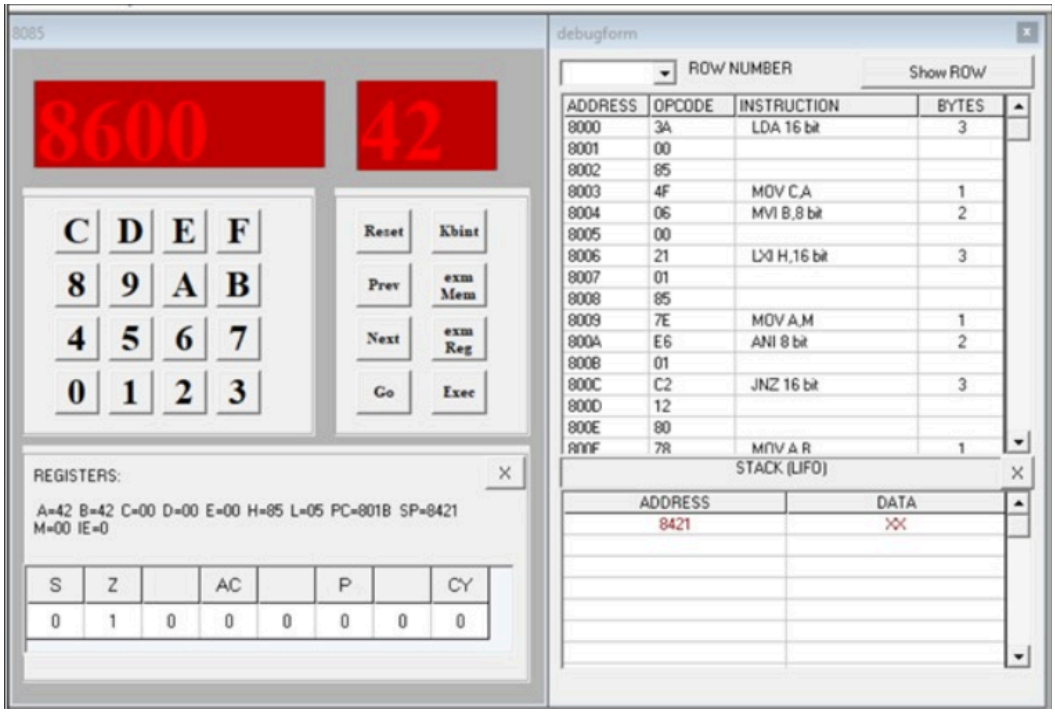
8085
8600
42
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=42 B=42 C=00 D=00 E=00 H=85 L=05 PC=801B SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3A LDA 16 bit 3
8001 00
8002 85
8003 4F MOV 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 7B MOV A,B 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 10.1

Experiment No.-11

Aim: 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	23
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

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

Output - [8600] – 01

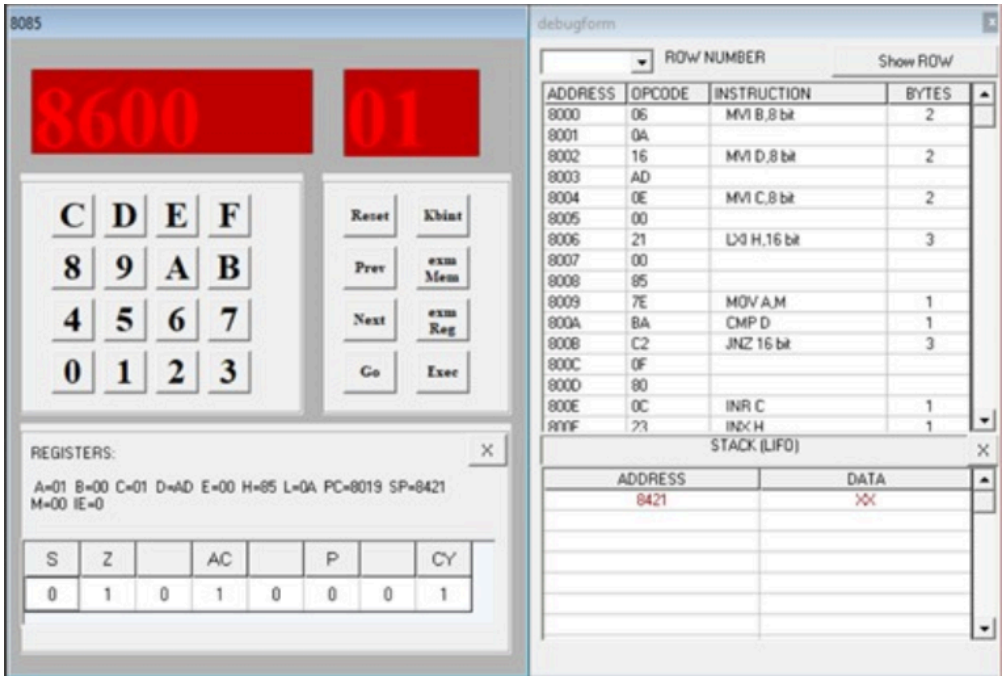
8085
8600
01
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=01 B=00 C=01 D=AD E=00 H=85 L=0A PC=8019 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 1 0 0 0 1
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 06 MVI B,8 bit 2
8001 0A
8002 16 MVI D,8 bit 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
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 11.1

Experiment No.-12

Aim: 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,E0,80
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

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

Output - [8600] – 05

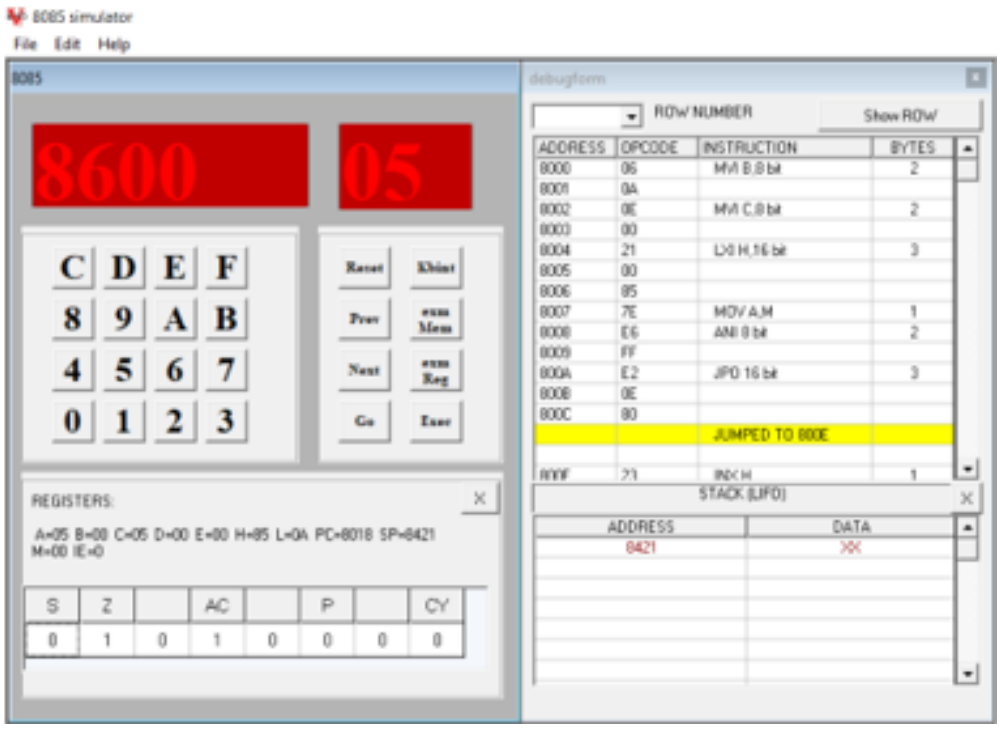
8085 simulator
File Edit Help
8085
8600
05
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=05 B=00 C=05 D=00 E=00 H=85 L=0A PC=8018 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 1 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 06 MVI B,8 bit 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 JPO 16 bit 3
800B 0E
800C 80
JUMPED TO 800E
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 12.1

Experiment No.-13

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

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

Table 13

Input - [8500] – 67

Output - [8600] – 43

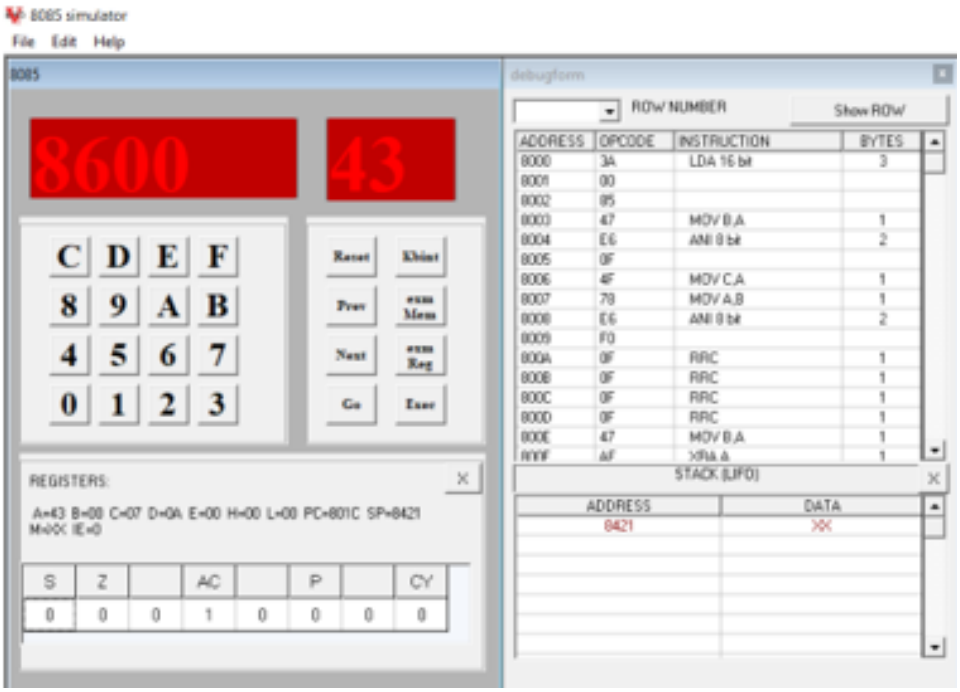
8085 simulator
File Edit Help
8085
8600
43
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Intrpt
Prev
exam Mem
Next
exam Reg
Go
Exec
REGISTERS:
A=43 B=00 C=07 D=0A E=00 H=00 L=00 PC=801C SP=8421
M=0X IE=0
S Z AC P CY
0 0 0 1 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3A LDA 16 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)
ADDRESS DATA
8421 XX

Fig 13.1

Experiment No.-14

Aim: Write an assembly language program for exchanging 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

Input - [8500] – 48, [8600] – 88

Output - [8500] – 88, [8600] – 48

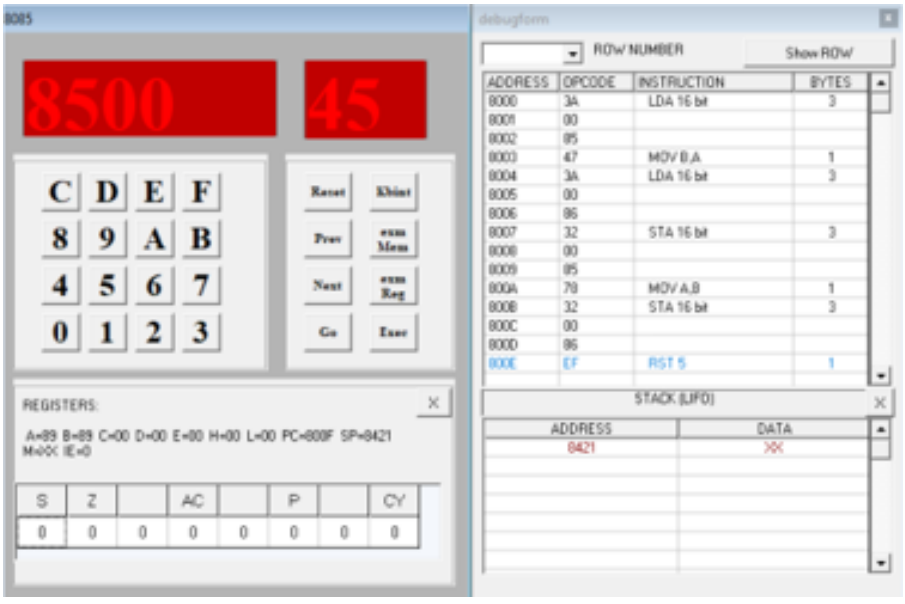

Fig 14.1

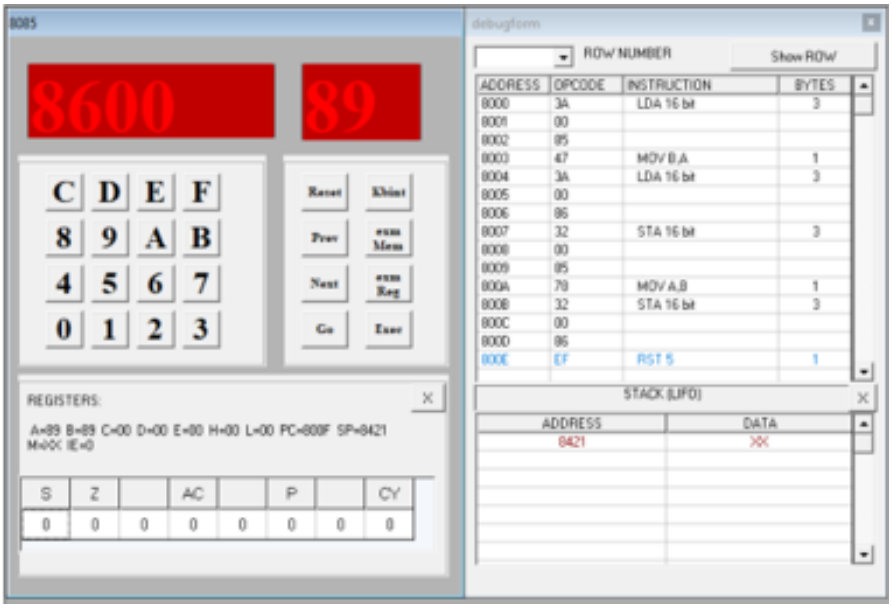
8085
8600
89
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Intrpt
Prev
Mem
Next
Reg
Go
Exec
REGISTERS:
A=89 B=89 C=00 D=00 E=00 H=00 L=00 PC=800F SP=8421
S Z AC P CY
0 0 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3A LDA 16 bit 3
8001 00
8002 85
8003 47 MOV B,A 1
8004 3A LDA 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)
ADDRESS DATA
8421 XX

Fig 14.2

Experiment No.-15

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

CODE	MEMORY LOCATION	OPCODE
MVI B, 09	8000,8001	06,09
LXI H, 8500H	8002,8003,8004	21,00,85
MOV A, M	8005	7E
INX H	8006	23
Back: CMP M	8007	BE
JNC Next	8008,8009,800A	D2,0C,80
MOV A, M	800B	7E
Next: INX H	800C	23
DCR B	800D	05
JNZ Back	800E,800F,8010	C2,07,80
STA 850AH	8011,8012.8013	32,0A,85
RST 5	8014	EF

Table 15

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

Output - [850A] – 0A

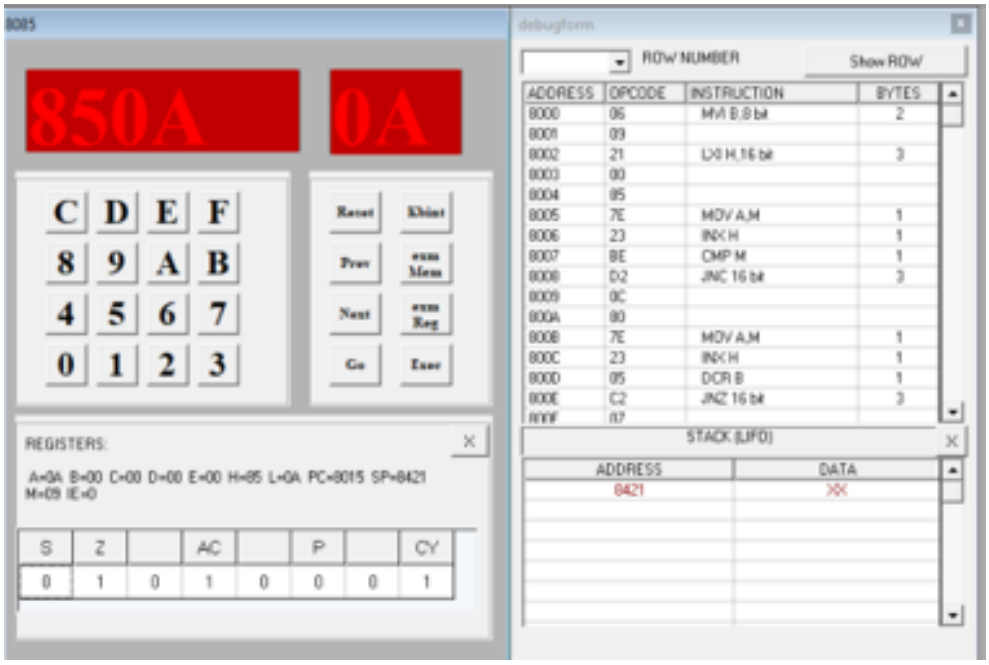
8085
850A
0A
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Intnt
Prev
exam Mem
Next
exam Reg
Go
Exec
REGISTERS:
A=0A B=00 C=00 D=00 E=00 H=85 L=0A PC=8015 SP=8421
M=09 IE=0
S Z AC P CY
0 1 0 1 0 0 0 1
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 06 MVI B,8 bit 2
8001 09
8002 21 LXI H,16 bit 3
8003 00
8004 85
8005 7E MOV A,M 1
8006 23 INX H 1
8007 BE CMP M 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
STACK (LIFO)
ADDRESS DATA
8421 XX

Fig 15.1