A
Practical Activity Report
Submitted for

MICROPROCESSOR-BASED SYSTEMS DESIGN (UCS617)

By

Submitted by :

Aadishri Soni (102083030)
Rahul Mishra (102083033)
Vipasha (102083057)
Cheshta (102083058)
Aprumpal Singh (102083062)

Submitted to :

Parminder Kaur

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY, PATIALA-147004,
PUNJAB
INDIA

Jan-June 2022

Contents

Experiment:1

Objective: Introduction to 8085 microprocessor kit and system specifications

Theory: 8085 is an 8-bit microprocessor designed by Intel in 1977 using NMOS technology. It has the following configuration –

- 8-bit data bus 16-bit address bus, which can address upto 64KB
- A 16-bit program counter
- A 16-bit stack pointer
- Six 8-bit registers arranged in pairs: BC, DE, HL
- Requires +5V supply to operate at 3.2 MHZ single phase clock

8085 consists of the following functional units:

Accumulator: It is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/STORE operations. It is connected to the internal data bus & ALU.

Arithmetic and logic unit: It performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

General purpose register: There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H & L. Each register can hold 8-bit data. These registers can work in pair to hold 16-bit data and their pairing combination is like B-C, D-E & H-L.

Program counter: It is a 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

Stack pointer: It is also a 16-bit register that works like a stack, which is always incremented/decremented by 2 during push & pop operations.

Temporary register: It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

Flag register: It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

Serial Input/output control: It controls the serial data communication by using these two instructions: SID (Serial input data) and SOD (Serial output data).

Address buffer and address-data buffer: The content stored in the stack pointer and program counter is loaded into the address buffer and address-data buffer to communicate with the CPU. The memory and I/O chips are connected to these buses; the CPU can exchange the desired data with the memory and I/O chips.

Address bus and data bus: Data bus carries the data to be stored. It is bidirectional, whereas the address bus carries the location to where it should be stored and it is unidirectional. It is used to transfer the data & Address I/O devices.

8085 is used in washing machines, microwave ovens, mobile phones, etc.

Steps To perform on the Simulator:

1. Press Reset

2. Press Examine Memory

3. Enter starting Address of the program

4. Press Next

5. Enter opcodes by subsequently pressing Next

6. Press Reset

7. Press Go

8. Enter Starting Address of the program to Compile

9. Press EXEC/FILL

10. Press Reset

11. Press Examine Memory

12. Enter Output Address

13. Press Next

Reference:

https://www.tutorialspoint.com/microprocessor/microprocessor_8085_architecture.htm

Experiment:2(i)

Objective: Write a program to store 8-bit data in one segment and then copy it to all registers.

Code:

MVI A, 48

MOV B, A

MOV C, A

MOV D, A

MOV E, A

MOV H, A

MOV L, A

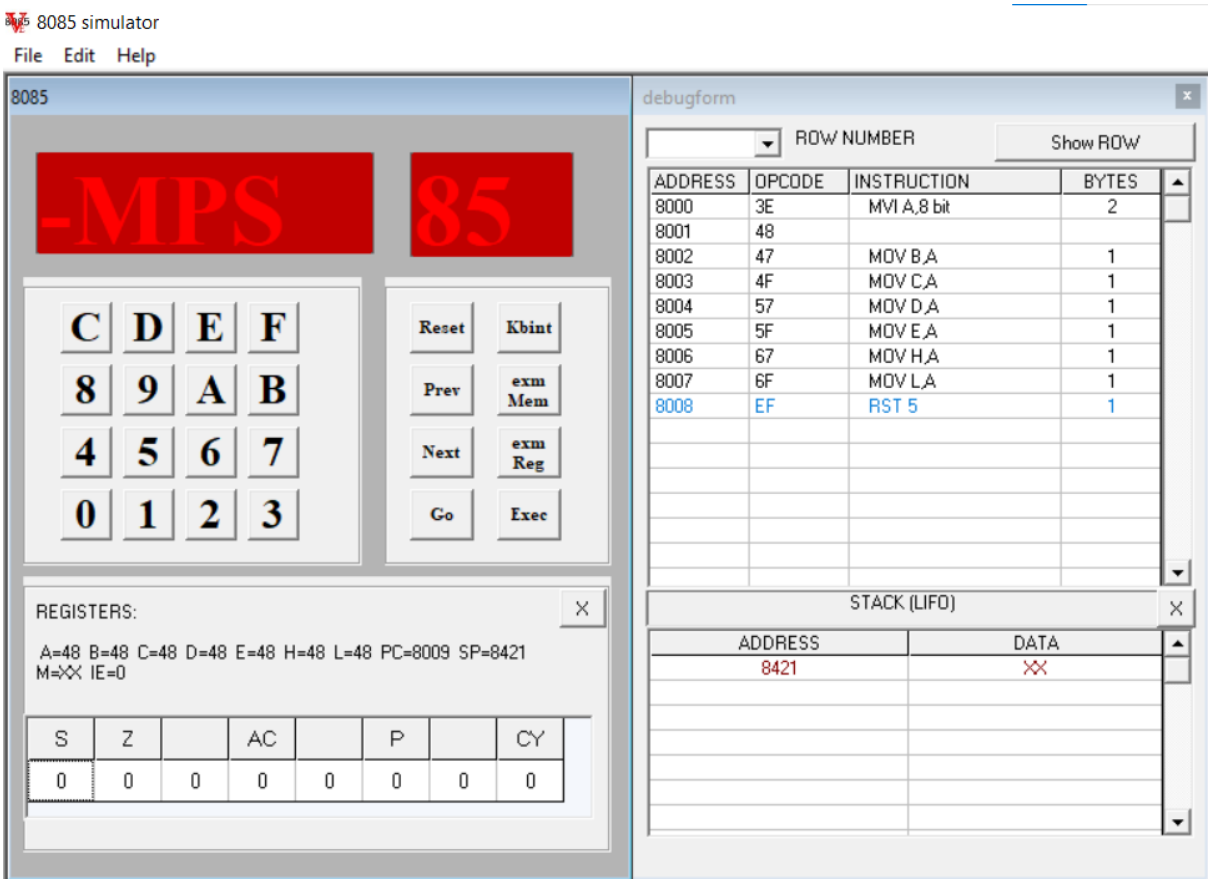
RST 5

Assembly Language Code:

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

INPUT A=48H OUTPUT B=48H C=48H D=48H E=48H H=48H L=48H

Output:



The number is copied to all the registers successfully, as seen in the output.

Experiment: 2(ii)

Objective: Write a program for addition of two 8-bit data numbers

Code:

MVI A, 48

MVI B, 48

ADD B

STA 9000

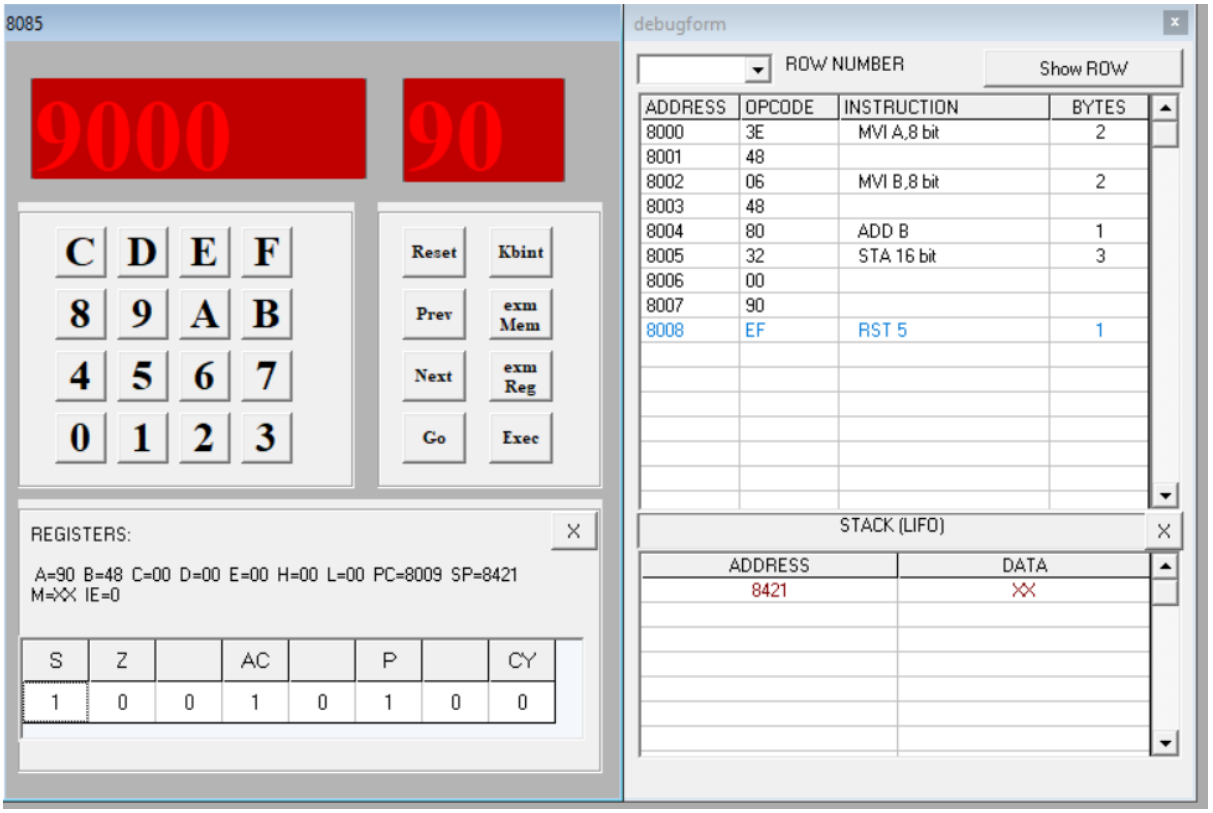
RST 5

Assembly Language Code:

Address	Instruction	OPCode
8000	MVI A, 48	3E
8001		48
8002	MVI B, 48	06
8003		48
8004	ADD B	80
8005	STA 9000	32
8006		00
8007		90
8008	RST 5	EF

INPUT A=48 B=48 OUTPUT [8500]=90

Output:



The result of addition is stored at the memory location 9000.

Experiment: 2(iii)

Objective: Write a program for addition of two 8-bit data numbers using direct and indirect addressing mode

Code:

Direct Addressing Mode:

LDA 8500

MOV B, A

LDA 8501

ADD B

STA 8502

RST 5

Assembly Language Code:

Address	Instruction	OPCode
8000	LDA 8500	3A
8001		00
8002		85
8003	MOV B, A	47
8004	LDA 8501	3A
8005		01
8006		85
8007	ADD B	80
8008	STA 8502	32
8009		02
800A		85
800B	RST 5	EF

INPUT [8500]=88 [8501]=88 OUTPUT [8502]=10

Indirect Addressing Mode:

LXI H,8500

MOV A, M

INX H

ADD M

INX H

MOV M, A

RST 5

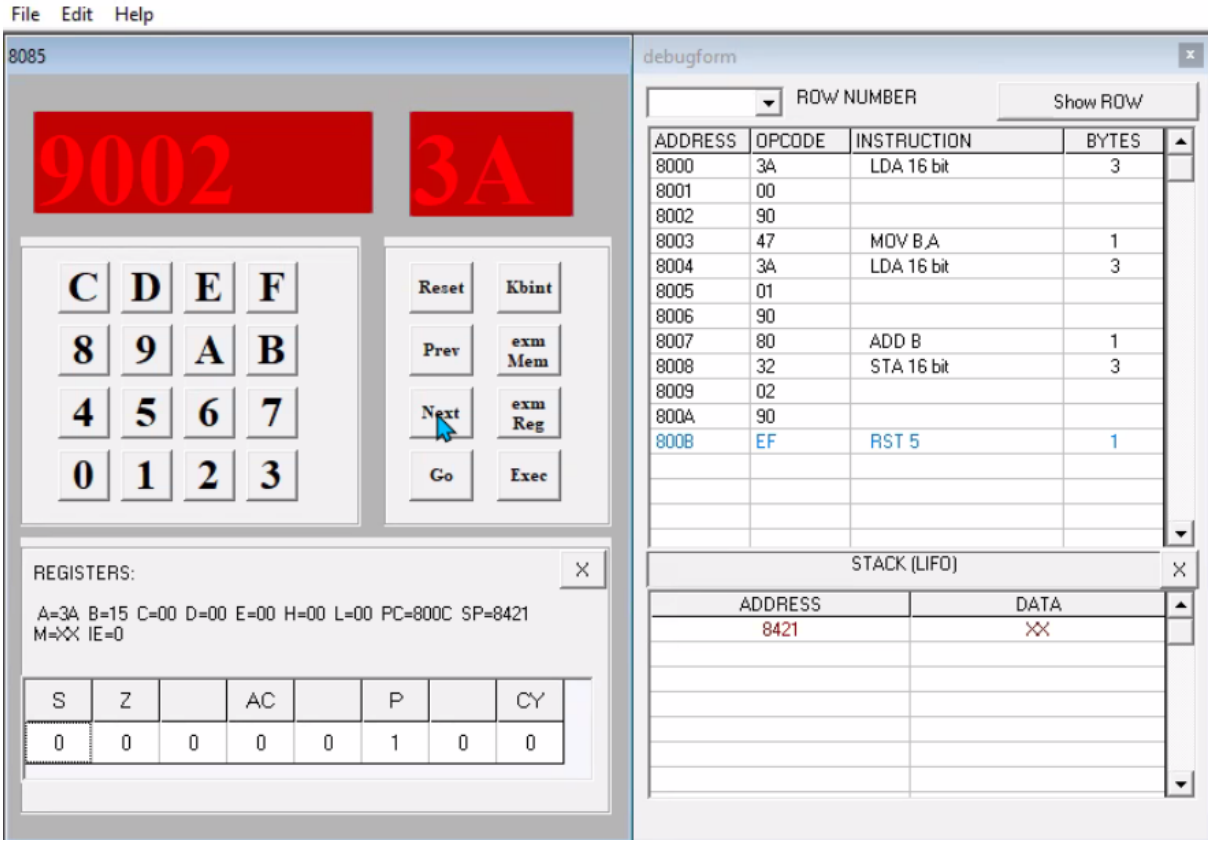
Assembly Language Code:

Address	Instruction	OpCode
8000	LXI H, 8500	21
8001		00
8002		85
8003	MOV A, M	7E
8004	INX H	23
8005	ADD M	86
8006	INX H	23
8007	MOV M, A	77
8008	RST 5	EF

Input: [9000]:08 , [9001]:32

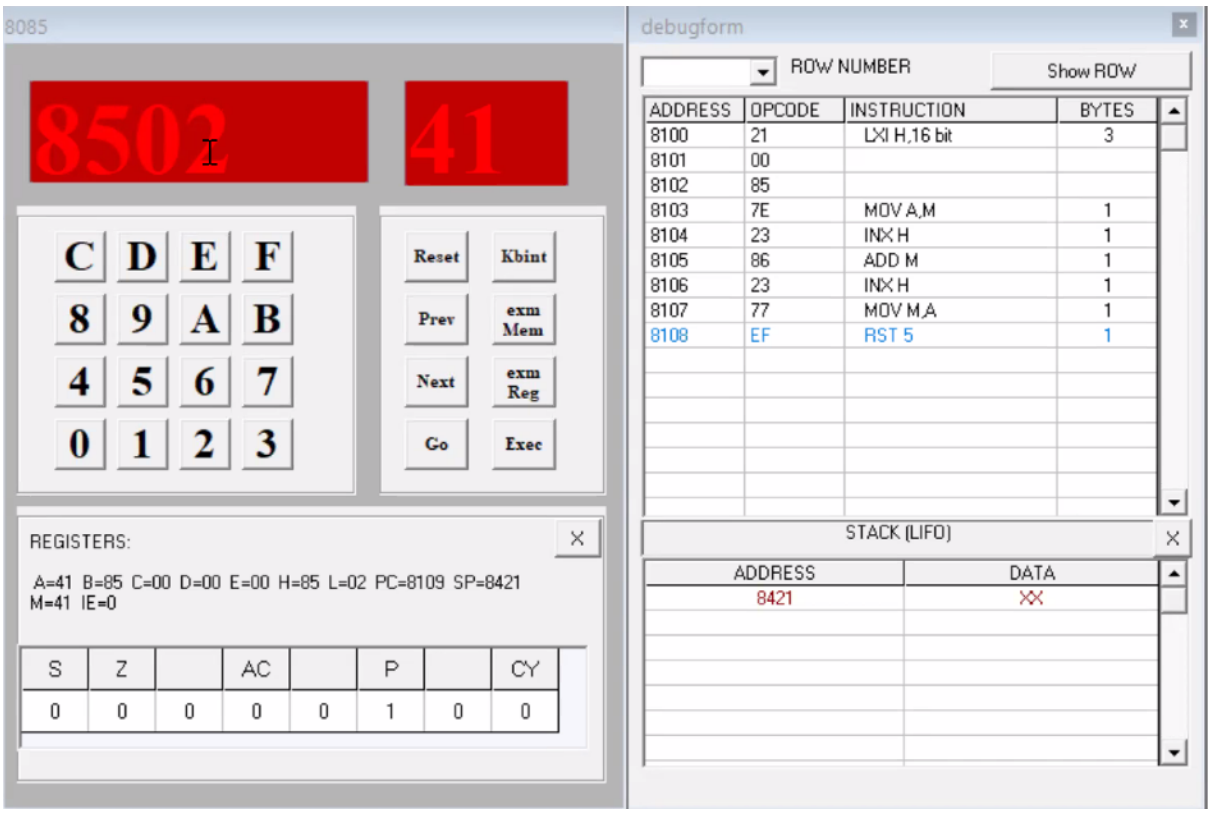
Output:

Direct Addressing Mode:



The result of addition is stored at the memory location 9002.

Indirect Addressing Mode:



The result of addition is stored at the memory location 8502.

Experiment: 2(iv)

Objective: Write a program for addition of two 16-bit data numbers using direct and indirect addressing mode

Code:

Direct Addressing:

LHLD 9000

XCHG

LHLD 9002

DAD D

SHLD 9004

RST5

Assembly Language Code:

Address	Instruction	OpCode
8000	LHLD 9000	2A
8001		00
8002		90
8003	XCHG	EB
8004	LHLD 9002	2A
8005		02
8006		90
8007	DAD D	19
8008	SHLD 9004	22
8009		04
800A		90
800B	RST5	EF

Input: [9000]- 48, [9001]-48, [9002]-48, [9003]-48

Indirect Addressing Mode:

LXI B, 9000

LDAX B

MOV D,A

INX B

LDAX B

ADD D

STA 9004

INX B

LDAX B

MOV D,A

INX B

LDAX B

ADC D

STA 9005

RST 5

Assembly Language Code:

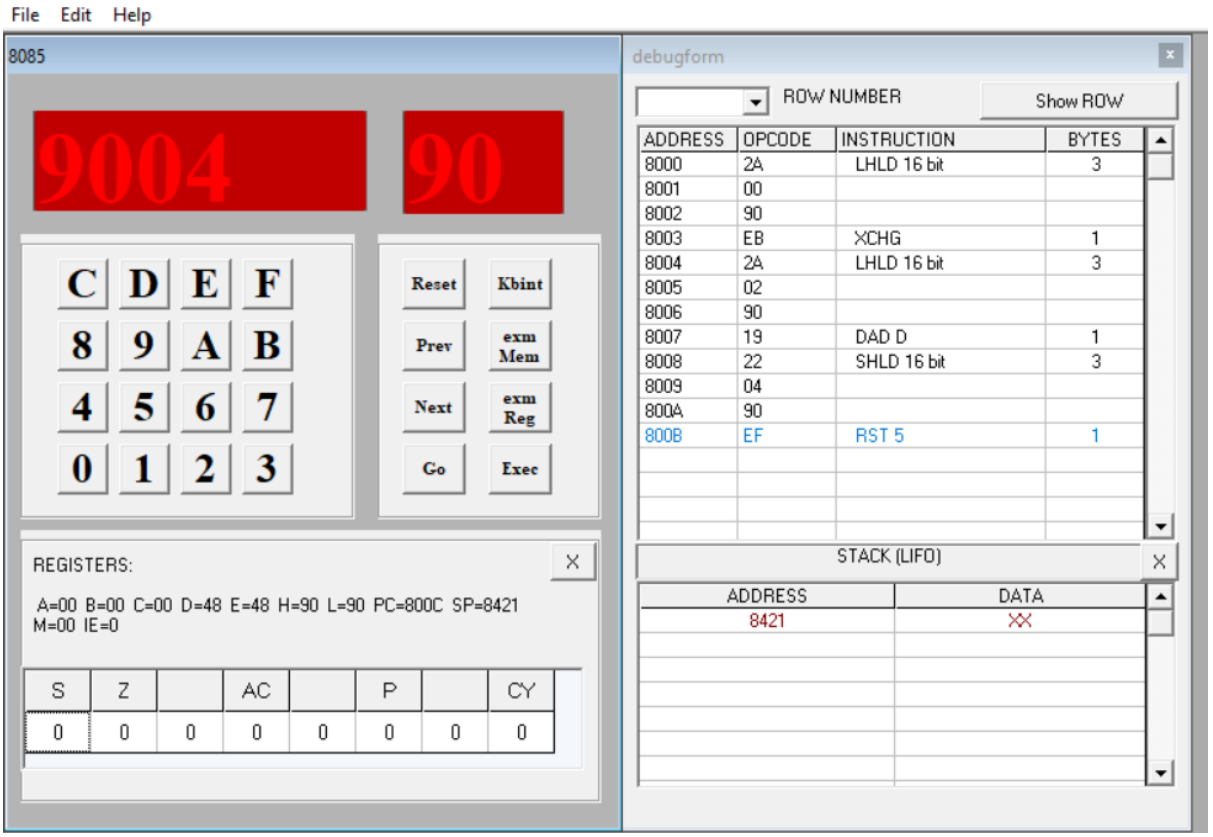
Address	Instruction	OpCode
8000	LXI B, 9000	01
8001		00
8002		90
8003	LDAX B	0A
8004	MOV D,A	57
8005	INX B	03
8006	LDAX B	0A
8007	ADD D	82
8008	STA 9004	32
8009		04
800A		90
800B	INX B	03
800C	LDAX B	0A
800D	MOV D,A	57
800E	INX B	03
800F	LDAX B	0A
8010	ADC D	8A
8011	STA 9005	32
8012		05

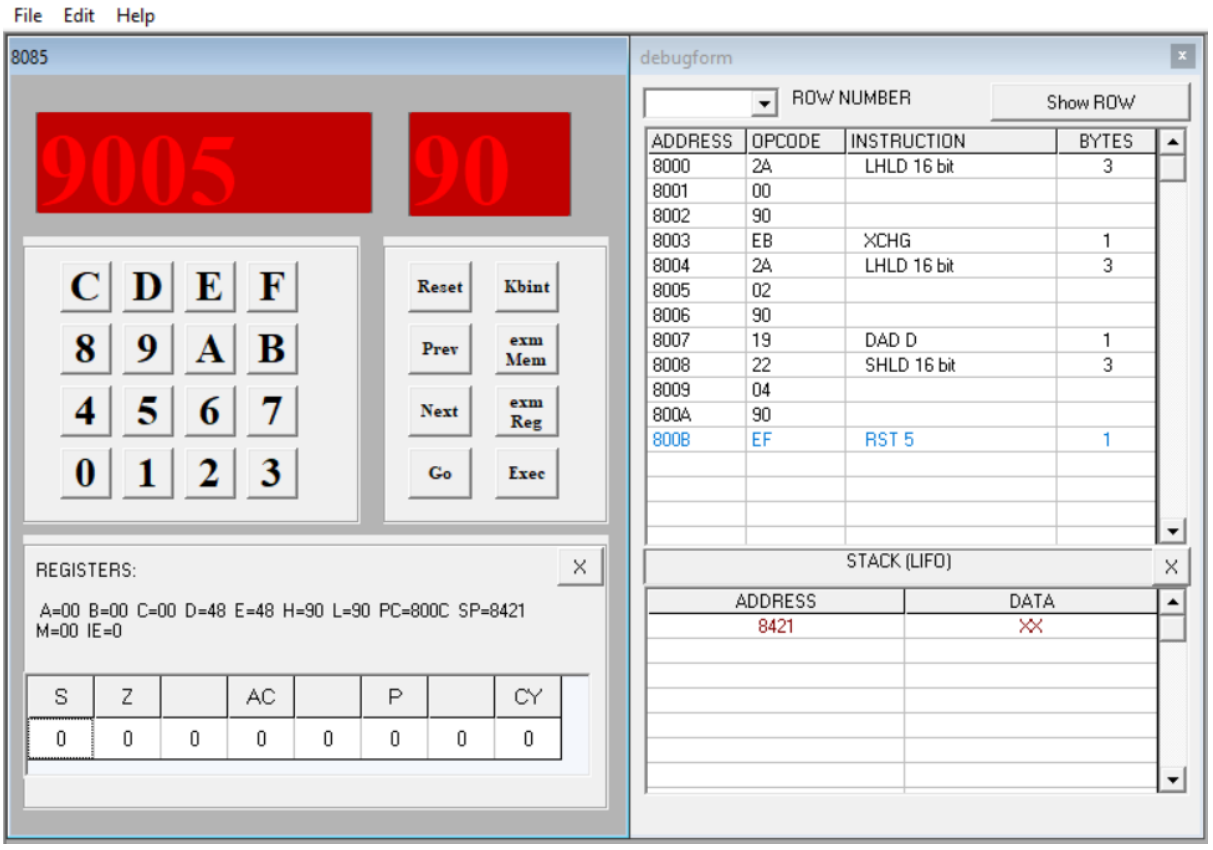
8013		90
8014	RST 5	EF

Input: [9000]- 34, [9001]-48, [9002]-54, [9003]-78

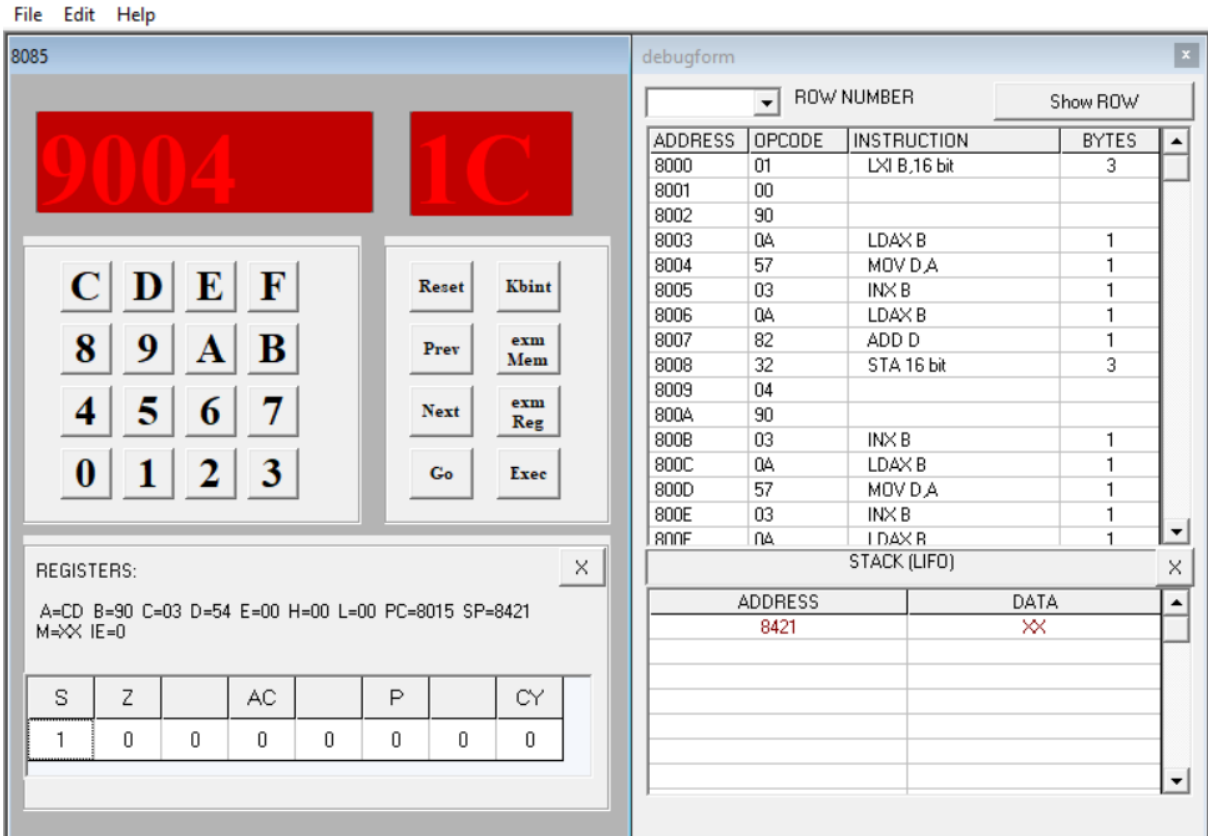
Output:

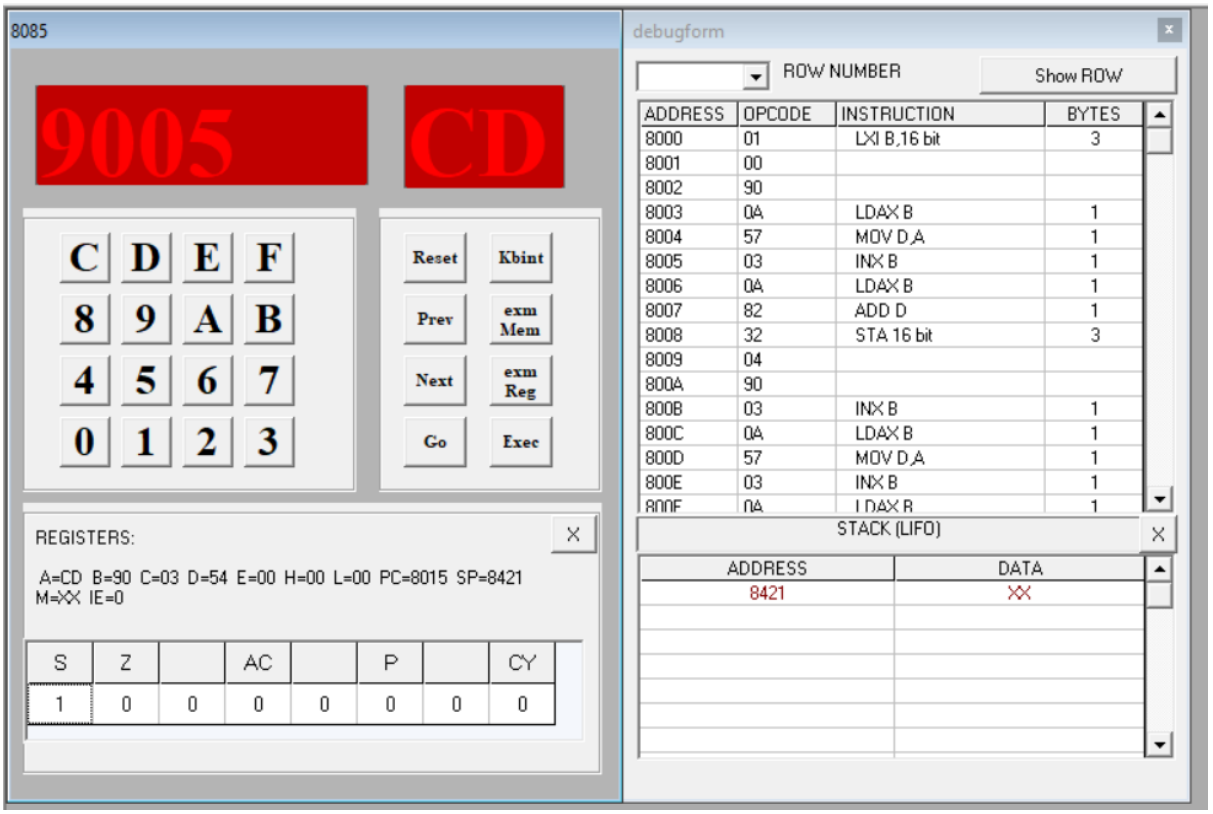
Direct Addressing Mode:





Indirect Addressing Mode:




8085
9005
CD
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:
X
A=CD B=90 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 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 01 LXI B,16 bit 3
8001 00
8002 90
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 90
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)
X
ADDRESS DATA
8421 XX

Experiment: 2(v)

Objective: Write a program for addition of two 8-bit data numbers using carry (JNC Addition).

Code:

MVI C, 00

LXI H, 8500

MOV A,M

INX H

ADD M

JNC NEXT

INR C

NEXT: INX H

MOV M,A

INX H

MOV M,C

RST 5

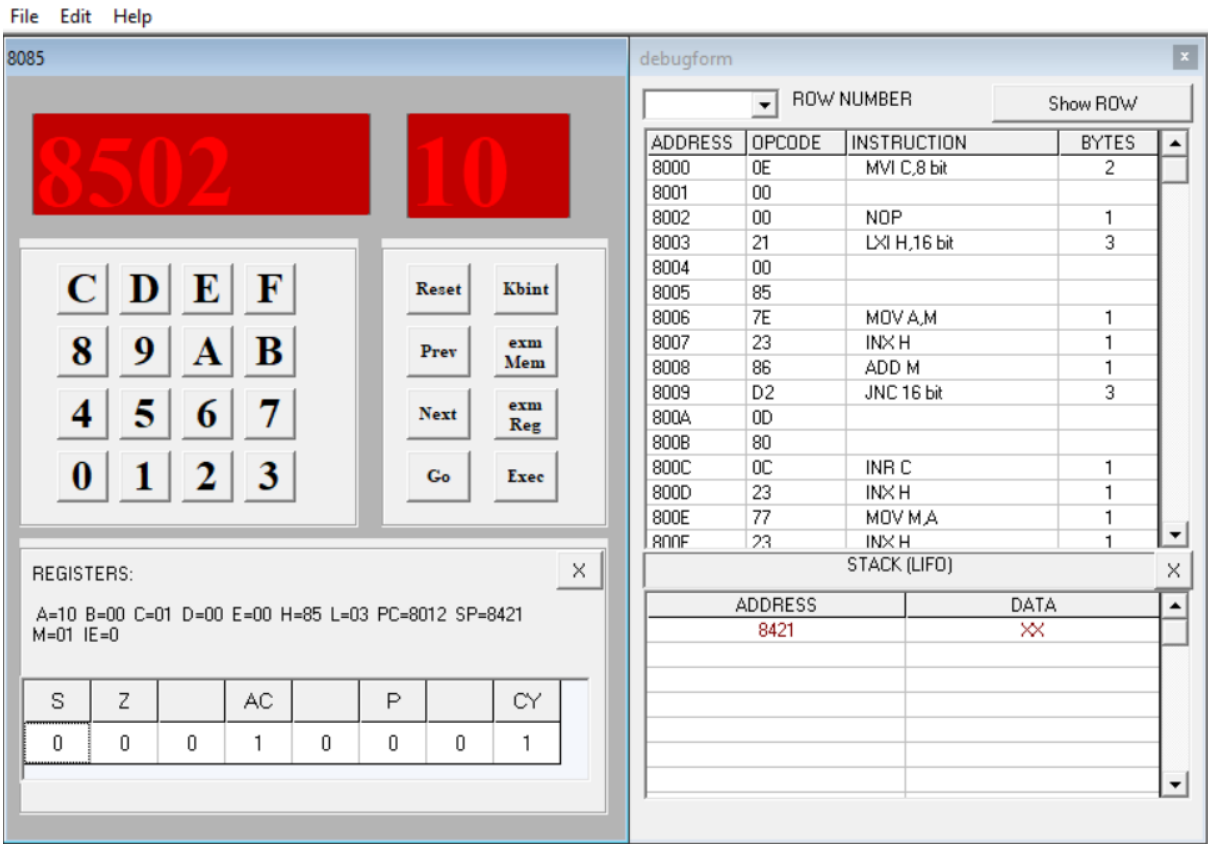
Assembly Language Code:

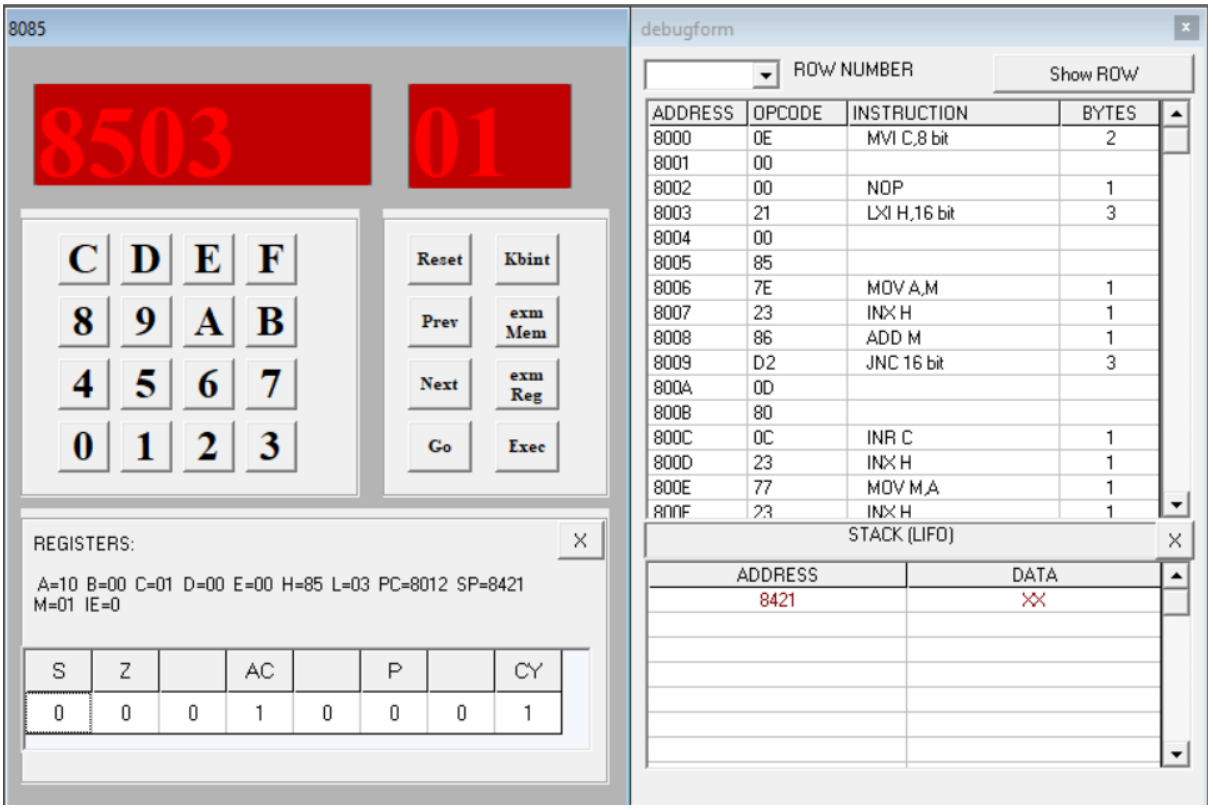
Address	Instruction	OPCode
8000	MVI C, 00	0E
8001		00
8002		00
8003	LXI H, 8500	21
8004		00
8005		85
8006	MOV A,M	7E
8007	INX H	23
8008	ADD M	86
8009	JNC Next	D2
800A		0D
800B		80
800C	INR C	0C

800D	Next INX H	23
800E	MOV M,A	77
800F	INX H	23
8010	MOV M,C	71
8011	RST 5	EF

Input: [8500]-88, [8501]-88

Output:




8085
8503
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=10 B=00 C=01 D=00 E=00 H=85 L=03 PC=8012 SP=8421
M=01 IE=0
S Z AC P CY
0 0 0 1 0 0 0 1
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 0E MVI C,8 bit 2
8001 00
8002 00 NOP 1
8003 21 LXI H,16 bit 3
8004 00
8005 85
8006 7E MOV A,M 1
8007 23 INX H 1
8008 86 ADD M 1
8009 D2 JNC 16 bit 3
800A 0D
800B 80
800C 0C INR C 1
800D 23 INX H 1
800E 77 MOV M,A 1
800F 23 INX H 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Experiment: 2(vi)

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

Code:

LDA 8500

CMA

STA 8501

INR A

STA 8502

RST 5

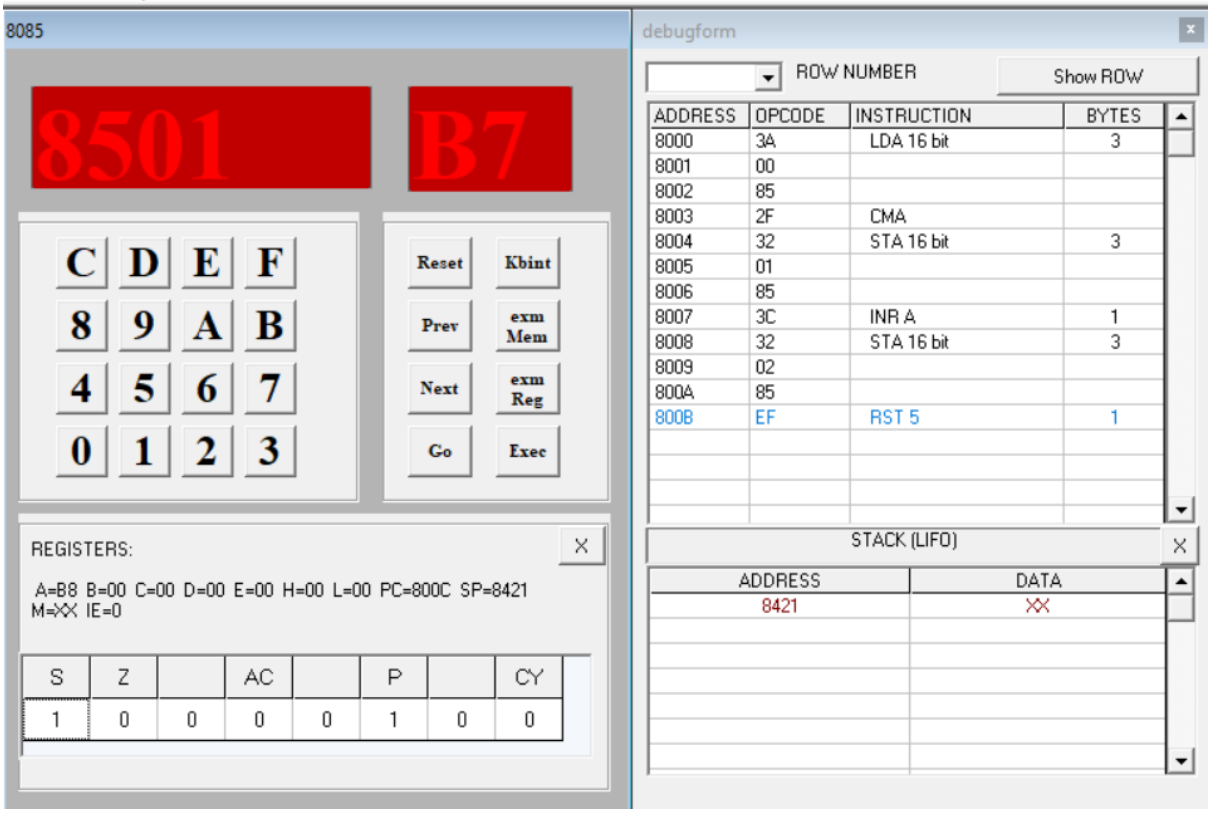
Assembly Language Code:

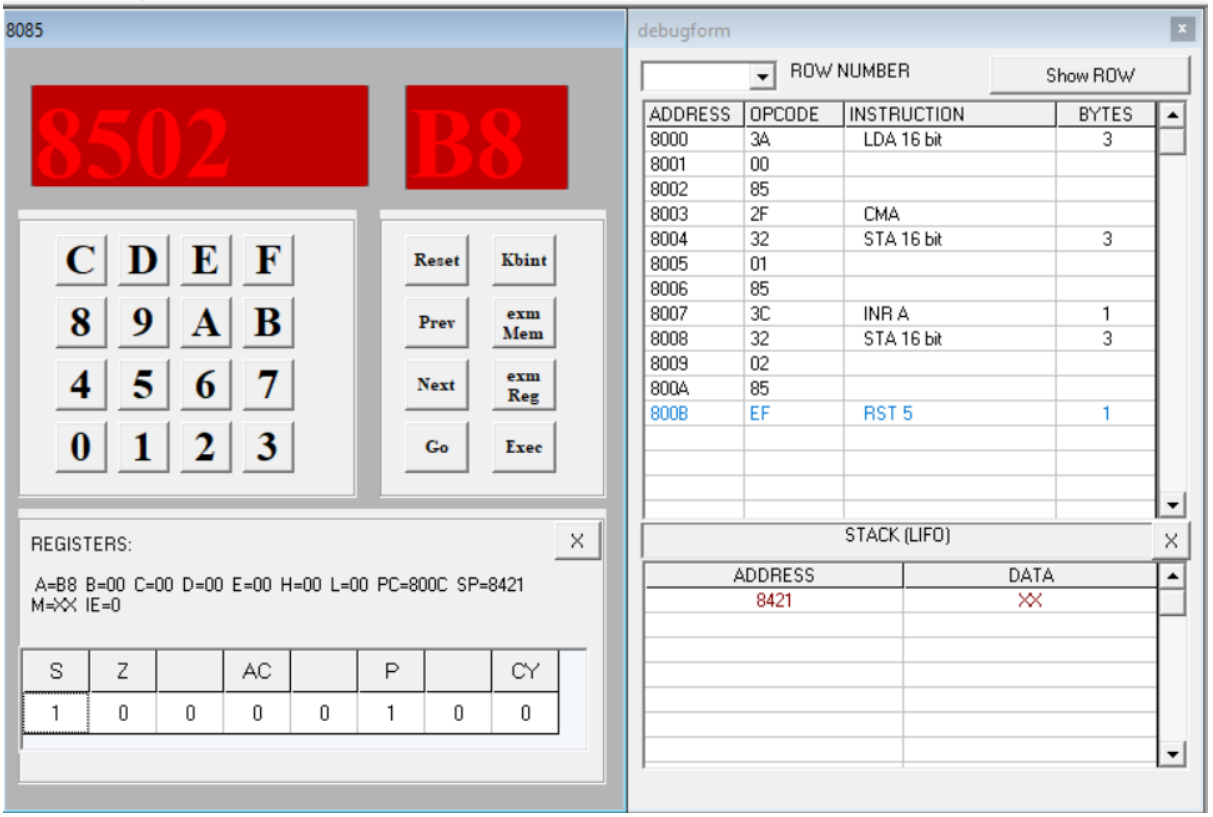
Address	Instruction	OPCode
8000	LDA 8500	3A
8001		00
8002		85
8003	CMA	2F
8004	STA, 8501	32
8005		01
8006		85
8007	INR A	3C
8008	STA, 8502	32
8009		02
800A		85
800B	RST 5	EF

Input: [8500]- 48

Output:

File Edit Help





Experiment: 3

Objective: Write a program for the sum of a series of numbers.

Code:

```
LDA 8200
MOV C,A
SUB A
LXI H,8201
BACK: ADD M
INX H
DCR C
JNZ BACK
STA 8300
RST 5
```

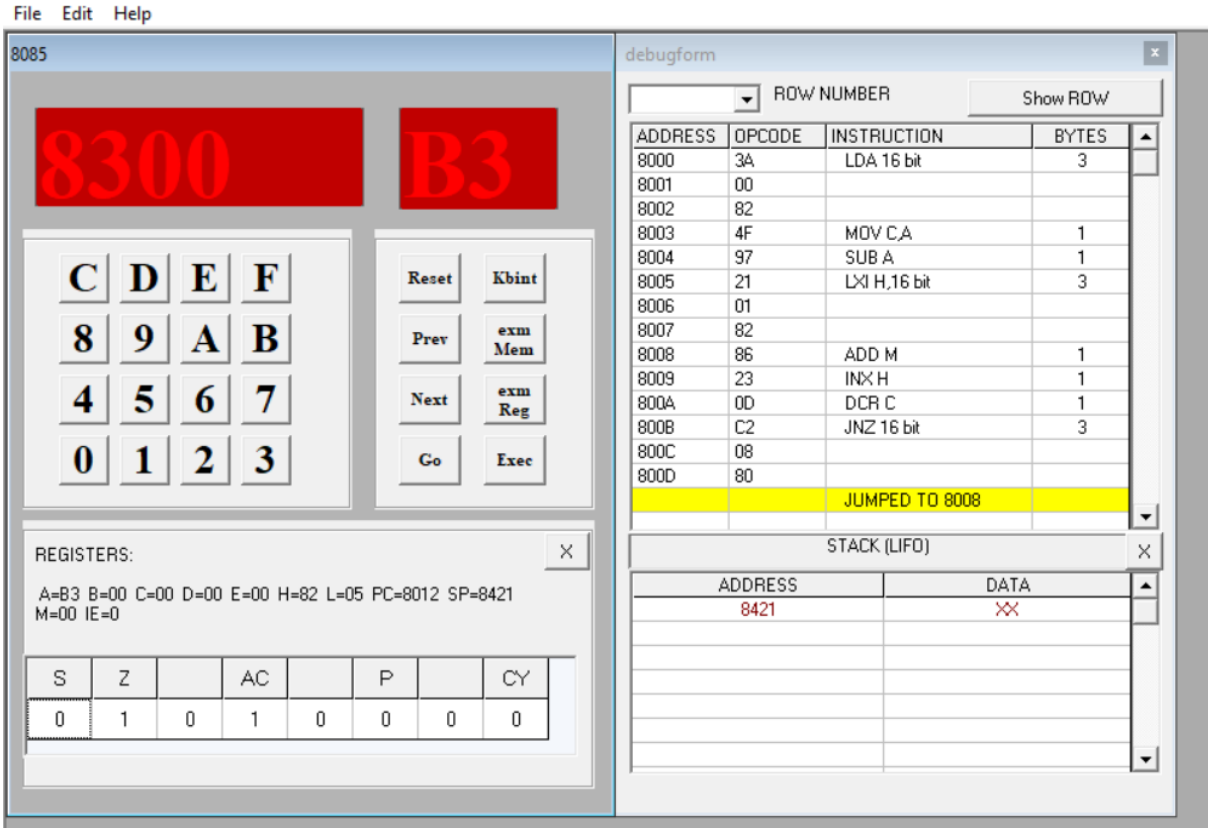
Assembly Language Code:

Address	Instruction	OPCode
8000	LDA 8200	3A
8001		00
8002		82
8003	MOV C,A	4F
8004	SUB A	97
8005	LXI H,8201	21
8006		01
8007		82
8008	BACK: ADD M	86
8009	INX H	23
800A	DCR C	0D
800B	JNZ BACK	C2
800C		08
800D		80
800E	STA 8300	32
800F		00
8010		83

8011	RST 5	EF

Input: [8200]-04, [8201]-9A, [820]-52, [8203]-89, [8204]-3E

Output:



Experiment: 4

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

Code:

```
MVI C, 0A
LXI H, 8500
LXI D, 8600
Back: MOV A,M
STAX D
INX H
INX D
DCR C
JNZ Back
RST 5
```

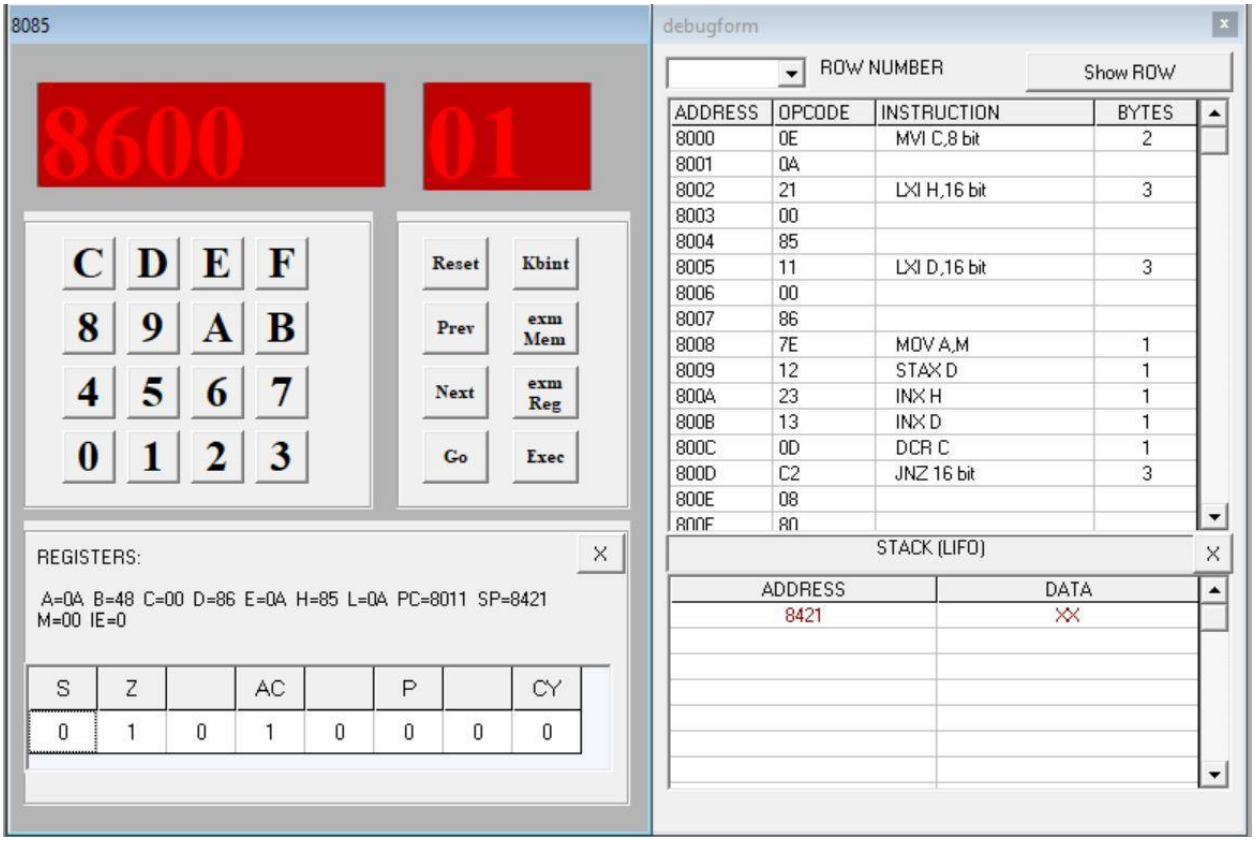
Assembly Language Code:

Address	Instruction	OPCODE
8000	MVI C, 0A	0E
8001		0A
8002	LXI H, 8500	21
8003		00
8004		85
8005	LXI D, 8600	11
8006		00
8007		86
8008	Back: MOV A,M	7E
8009	STAX D	12
800A	INX H	23
800B	INX D	13
800C	DCR C	0D

800D	JNZ Back	C2
800E		08
800F		80
8010	RST 5	EF

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



Experiment: 5

Objective: Write a program for Multiply two 8-bit numbers.
Code:

```
LDA 8500
MOV E, A
MVI D, 00
LDA 8501
MOV C, A
LXI H, 0000
Back: DAD D
DCR C
JNZ Back
SHLD 8600
RST 5
```

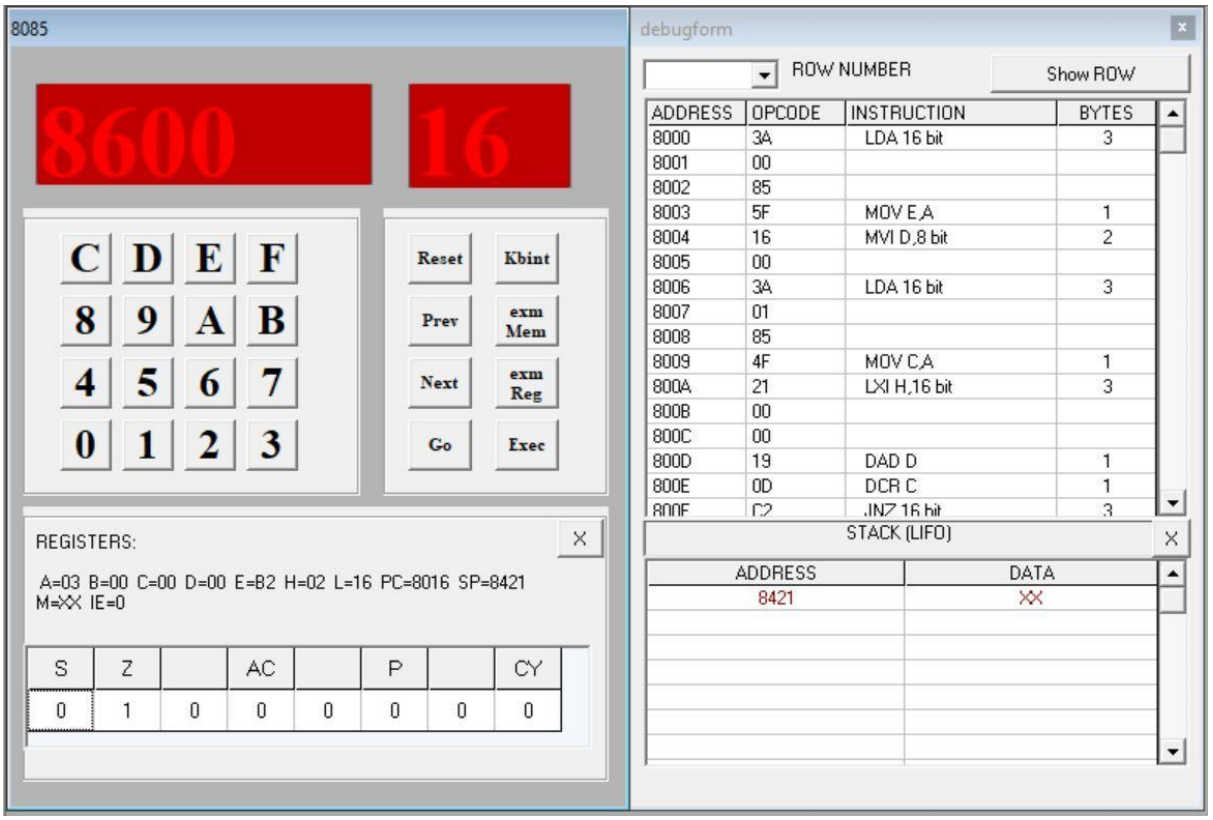
Assembly Language Code:

Address	Instruction	OPCODE
8000	LDA, 8500	3A
8001		00
8002		85
8003	MOV E, A	5F
8004	MVI D, 00	16
8005		00
8006	LDA, 8501	3A
8007		01
8008		85
8009	MOV C, A	4F
800A	LXI H, 0000	21
800B		00
800C		00

800D	Back: DAD D	19
800E	DCR C	0D
800F	JNZ Back	C2
8010		0D
8011		80
8012	SHLD, 8600	22
8013		00
8014		86
8015	RST 5	EF

Input: [8500] – B2 [8501] – 03

Output: [8600] – 16 [8601] – 02



Experiment: 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 addresses.

Code:

```
MVI C, 00
MVI B, 09
LXI H, 8500
MOV A, M
Back: INX H
ADD M
JNC Next
INR C
Next: DCR B
JNZ Back
INX H
MOV M, A
INX H
MOV M, C
RST 5
```

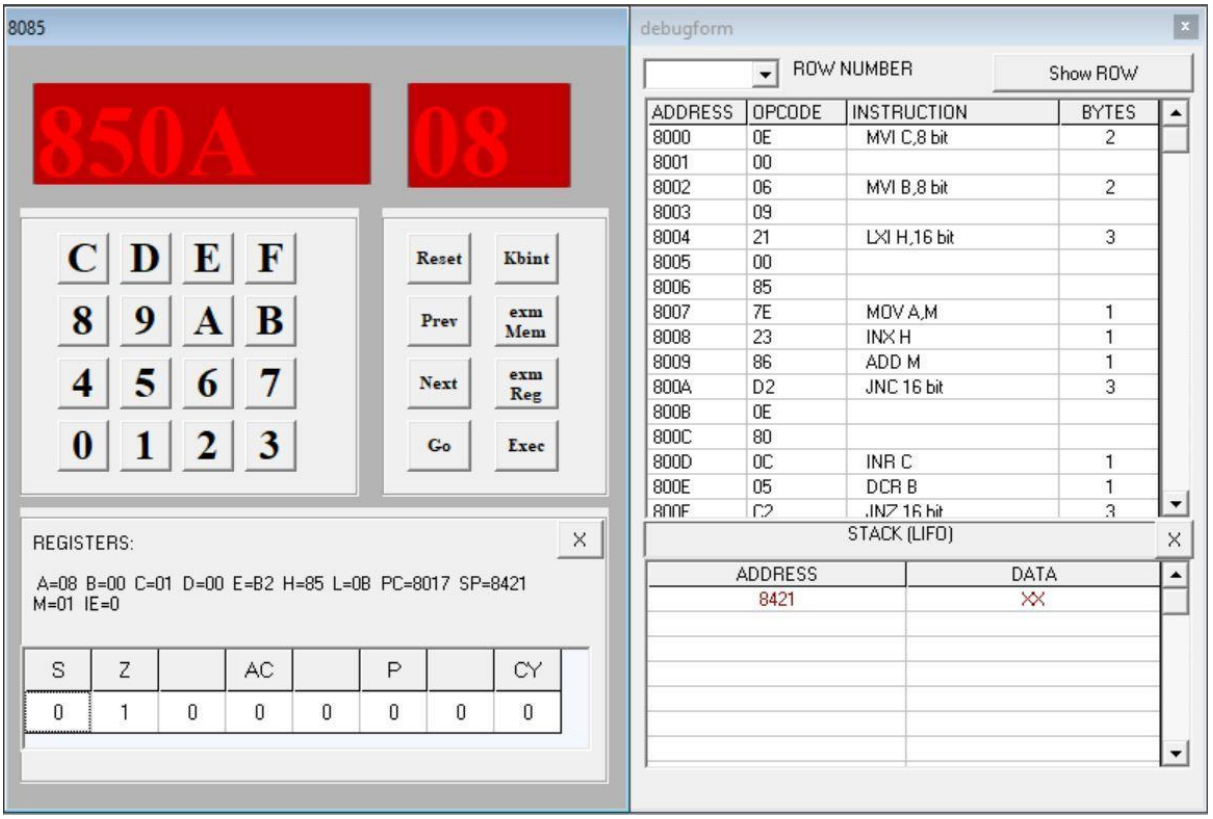
Assembly Language Code:

Address	Instruction	OPCODE
8000	MVI C, 00	0E
8001		00
8002	MVI B, 09	06
8003		09
8004	LXI H, 8500	21
8005		00
8006		85
8007	MOV A,M	7E
8008	Back: INX H	23
8009	ADD M	86

800A	JNC Next	D2
800B		0E
800C		80
800D	INR C	0C
800E	Next: DCR B	05
800F	JNZ Back	C2
8010		08
8011		80
8012	INX H	23
8013	MOV M,A	77
8014	INX H	23
8015	MOV M,C	71
8016	RST 5	EF

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


8085
850A
08
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=08 B=00 C=01 D=00 E=B2 H=85 L=0B PC=8017 SP=8421
M=01 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 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
STACK (LIFO)
ADDRESS DATA
8421 XX

Experiment: 7

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

Code:

```
LDA 8200
MOV C, A
MVI B, 00
LXI H, 8201
BACK: MOV A, M
ANI 80
JZ Skip
INR B
Skip: INX H
DCR C
JNZ BACK
MOV A, B
STA 8300
RST 5
```

Assembly Language Code:

Address	Instruction	OPCODE
8000	LDA 8200	3A
8001		00
8002		82
8003	MOV C, A	4F
8004	MVI B, 00	06
8005		00
8006	LXI H, 8201	21
8007		01
8008		82
8009	BACK: MOV A, M	7E
800A	ANI 80	E6

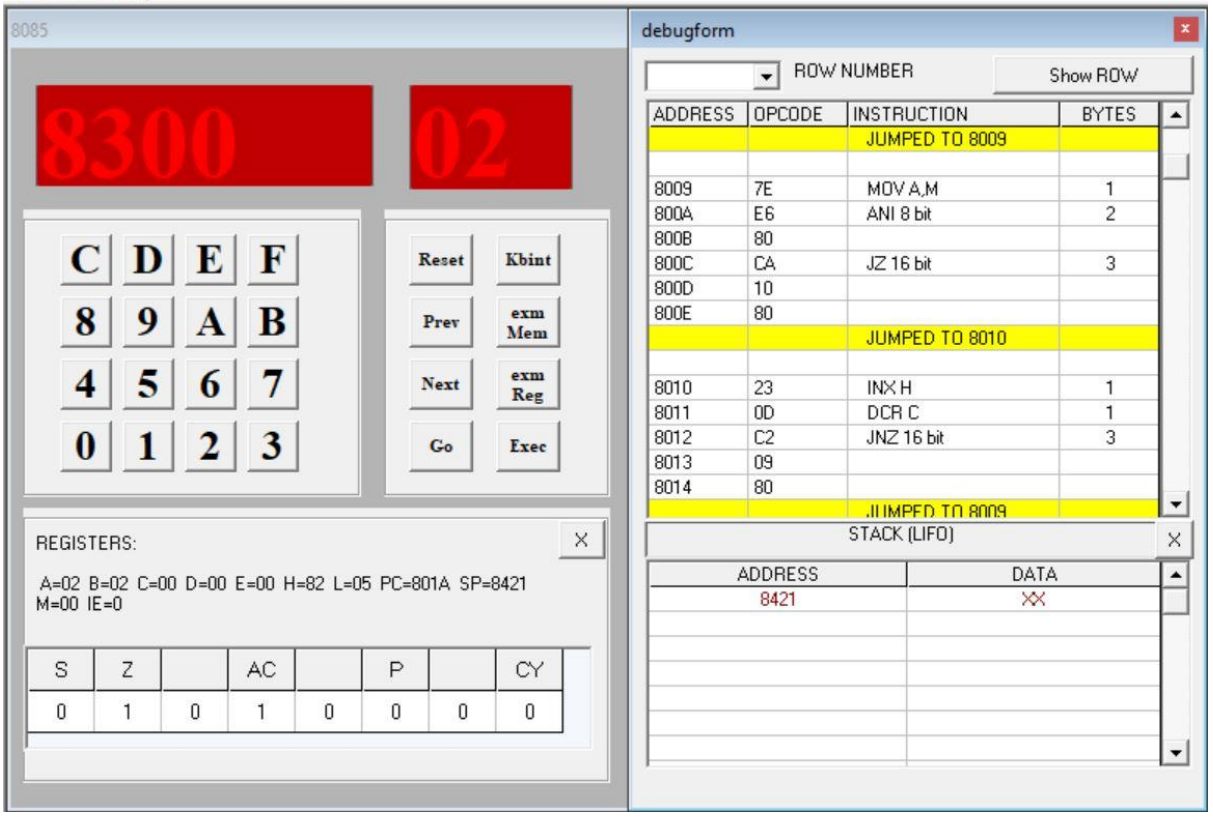
800B		80
800C	JZ Skip	CA
800D		10
800E		80
800F	INR B	04
8010	Skip: INX H	23
8011	DCR C	0D
8012	JNZ BACK	C2
8013		09
8014		80
8015	MOV A, B	78
8016	STA 8300	32
8017		00
8018		83
8019	RST 5	EF

Input: [8200]=04 [8201]=56 [8202]=A9 [8203]=73 [8204]=82

Output: [8300]=02

8085 simulator

File Edit Help



Experiment: 8

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

```
LDA 8500
MVI B,00
MVI D,08
Loop1: RLC
JNC Loop2
INR D
Loop2: DCR B
JNZ Loop1
MOV A,D
STA 9000
RST 5
```

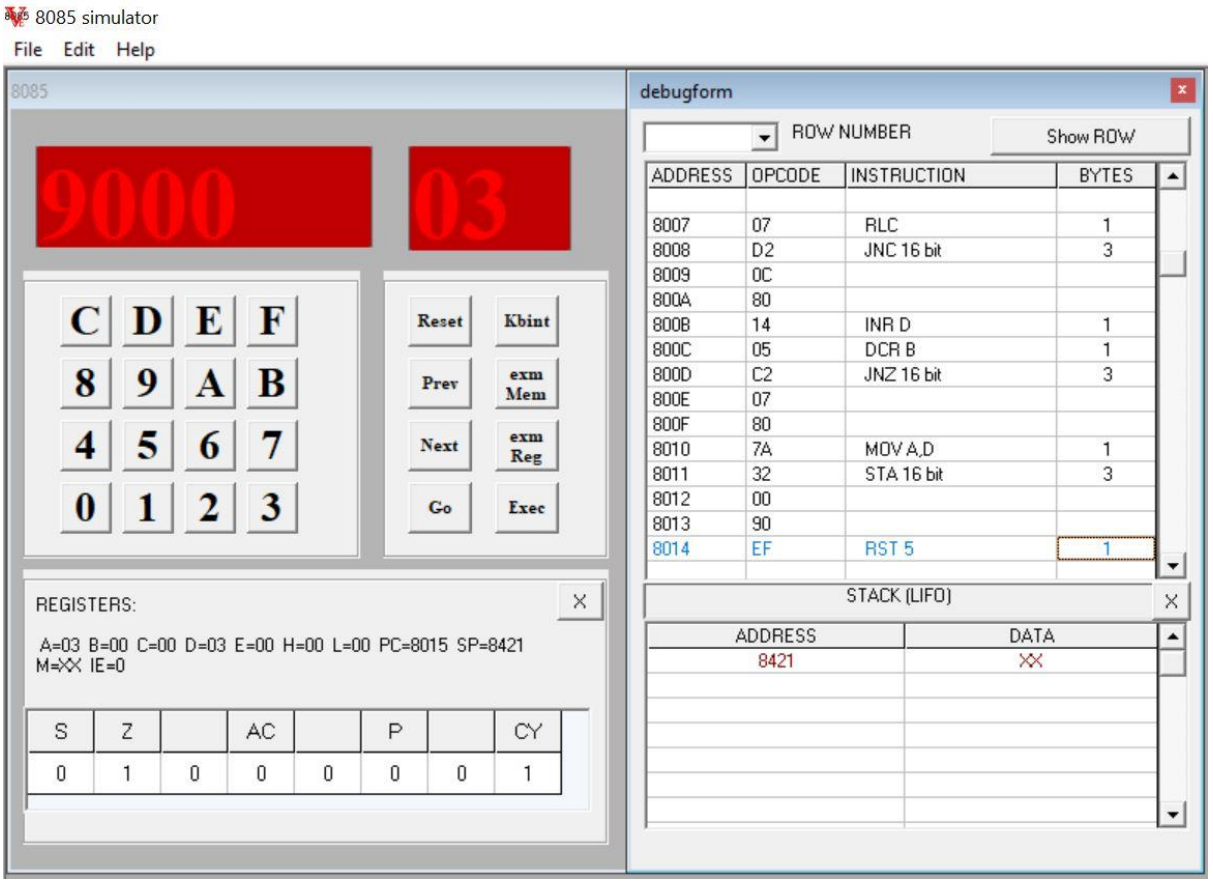
Assembly Language Code:

Address	**Instruction**	**OPCODE**
8000	LDA 8500	3A
8001		00
8002		85
8003	MVI B,00	06
8004		00
8005	MVI D,08	16
8006		08
8007	Loop1: RLC	07
8008	JNC Loop2	D2
8009		0C
800A		80
800B	INR D	14
800C	Loop2: DCR B	05

800D	JNZ Loop1	C2
800E		07
800F		80
8010	MOV A,D	7A
8011	STA 9000	32
8012		00
8013		90
8014	RST 5	EF

Input: [8500]=25

Output: [9000]=03



Experiment: 9

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

Code:

```
LXI H, 8500
MOV C, M
DCR C
Repeat: MOV D, C
LXI H, 8501
Loop: MOV A, M
INX H
CMP M
JC Skip
MOB B, M
MOV M, A
DCX H
MOV M, B
INX H
Skip: DCR D
JNZ Loop
DCR C
JNZ Repeat
RST 5
```

Assembly Language Code:

Address	Instruction	OPCODE
8000	LXI H, 8500	21
8001		00
8002		85
8003	MOV C, M	4E
8004	DCR C	0D
8005	Repeat: MOV D, C	51
8006	LXI H, 8501	21
8007		01

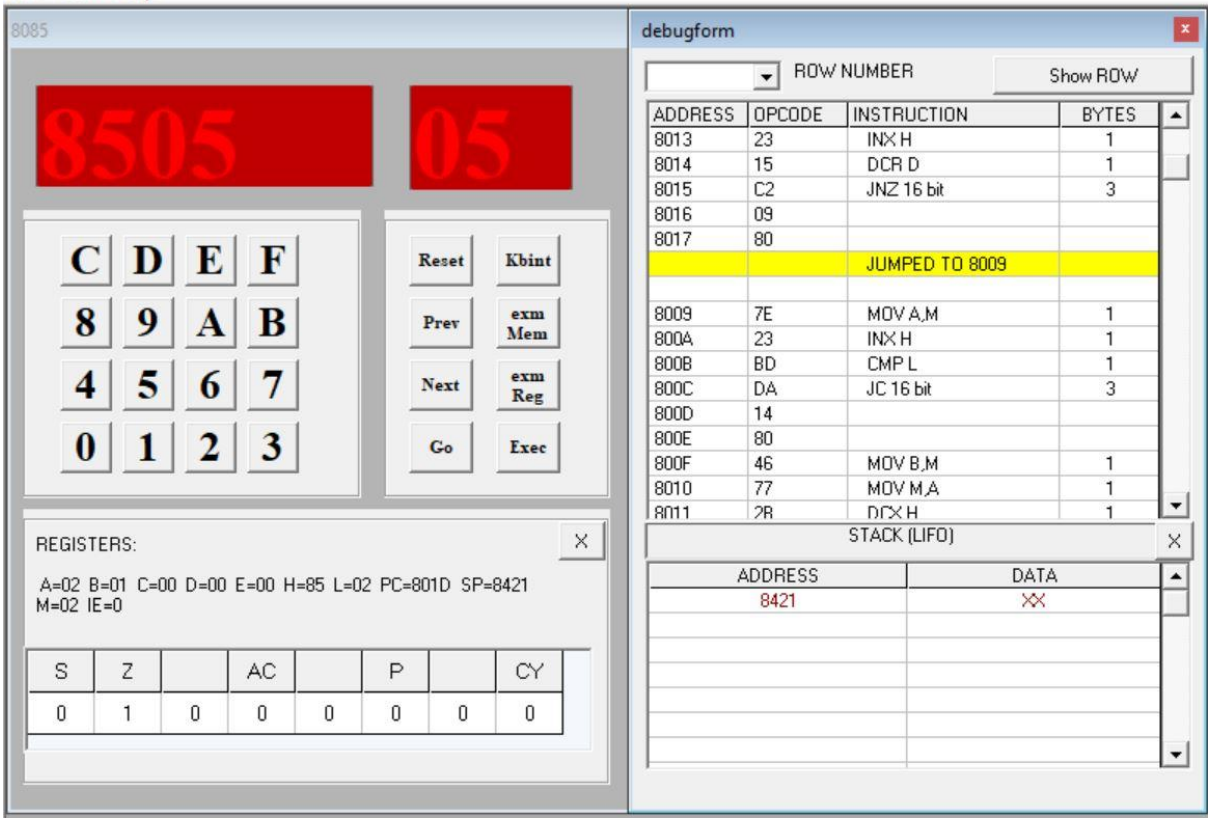
8008		85
8009	Loop: MOV A, M	7E
800A	INX H	23
800B	CMP M	BD
800C	JC Skip	DA
800D		14
800E		80
800F	MOB B, M	46
8010	MOV M, A	77
8011	DCX H	2B
8012	MOV M, B	70
8013	INX H	23
8014	Skip: DCR D	15
8015	JNZ Loop	C2
8016		09
8017		80
8018	DCR C	0D
8019	JNZ Repeat	C2
801A		05
801B		80
801C	RST 5	EF

Input: [8200]=05 [8201]=05 [8202]=04 [8203]=03 [8204]=02 [8205]=01

Output: [8200]=05 [8201]=01 [8202]=02 [8203]=03 [8204]=04 [8205]=05

8085 simulator

File Edit Help



Experiment: 10

Objective: Calculate the sum of a series of even numbers.

Code:

LDA 8200

MOV C, A

MVI B, 00

LXI H, 820

BACK: MOV A, M

ANI 01

JNZ SKIP

MOV A, B

ADD M

MOV B, A

SKIP INX H

DCR C

JNZ BACK

STA 820A

RST 5

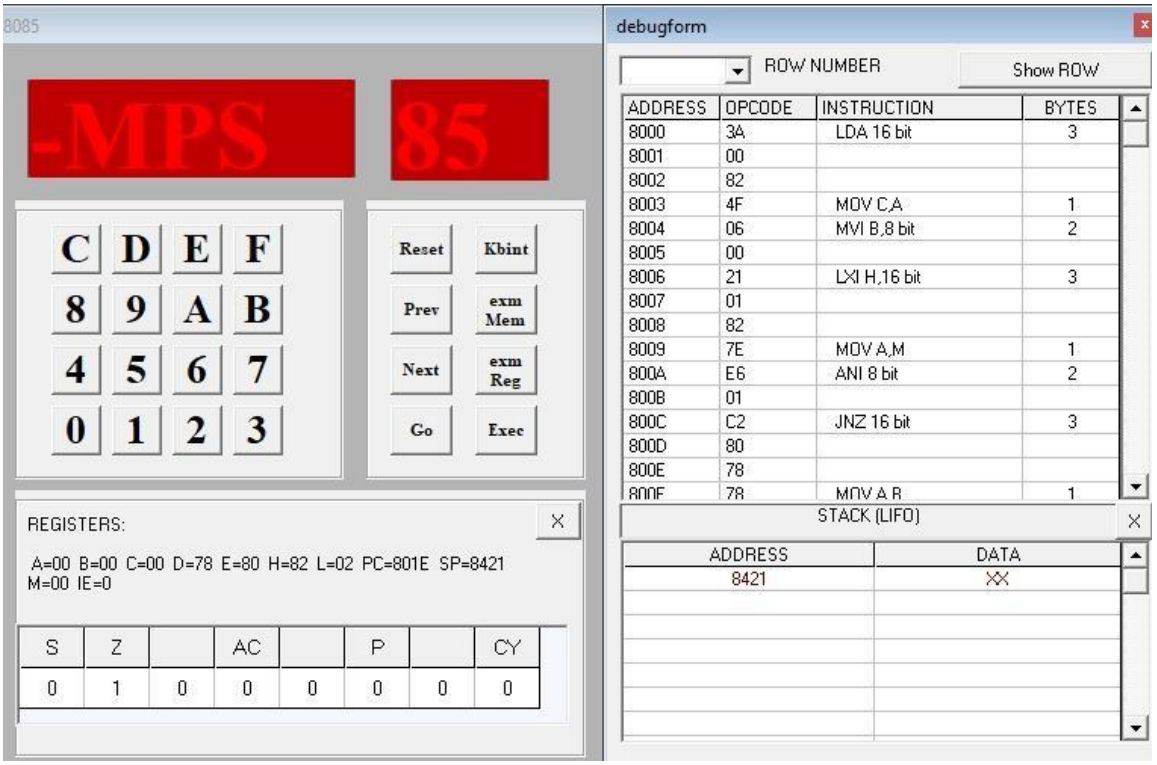
Assembly Language Code:

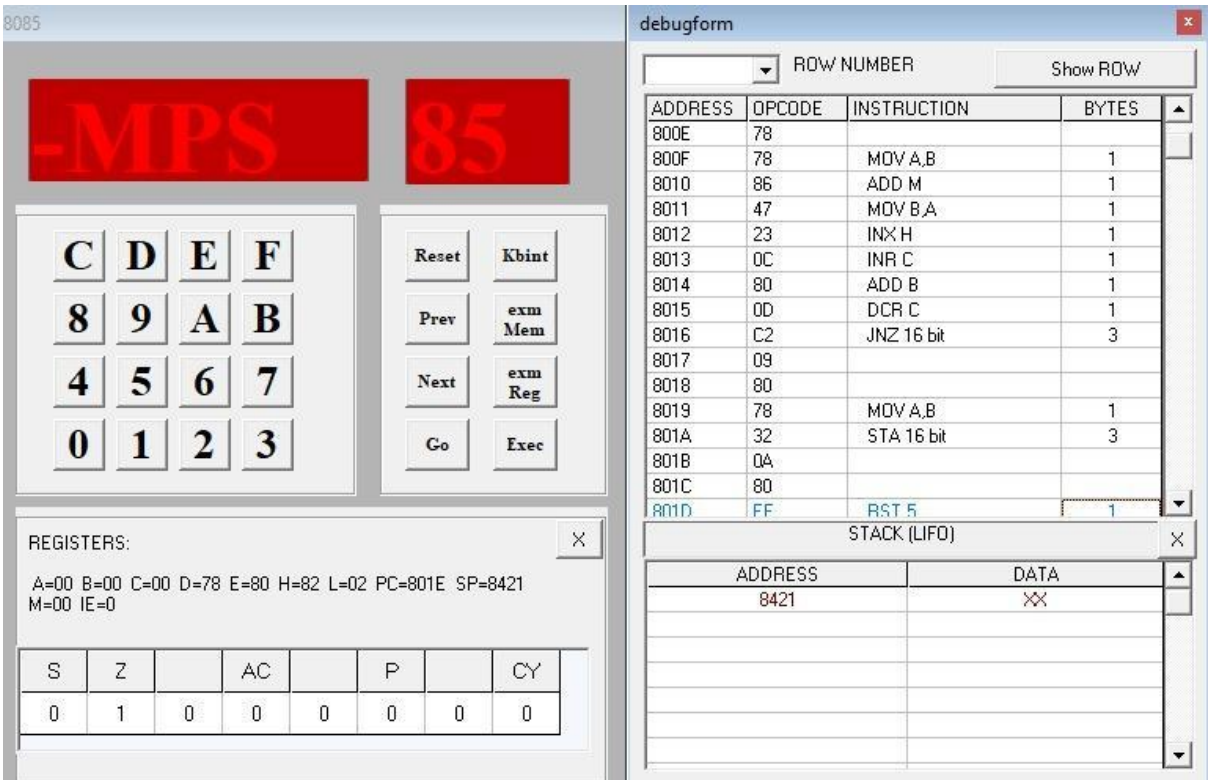
ADDRESS	INSTRUCTION	OPCODE
8000	LDA 16 bit	3A
8001		00
8002		82
8003	MOV C, A	4F
8004	MVI B ,8bit	06
8005		0
8006	LXI H, 16 bit	21
8007		01
8008		82
8009	MOV A,M	7E
800A	ANI 8 bit	E6

800B		01
800C	JNZ 16 bit	C2
800D		80
800E		78
800F	MOV A,B	78
8010	ADD M	86
ADDRESS	INSTRUCTION	OPCODE
8011	MOV B, A	47
8012	INX H	23
8013	INR C	0C
8014	ADD B	80
8015	DCR C	0D
8016	JNZ 16 bit	C2
8017		09
8018		80
8019	MOV A,B	78
801A	STA 16 bit	32
801B		0A
801C		80
801D	RST 5	FF

INPUT [9500]=04 [9501]=20 [9502]=15 [9503]=13 [9504]=22 **OUTPUT** [9510]=42

Output:


8085
-MPS
85
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=00 B=00 C=00 D=78 E=80 H=82 L=02 PC=801E 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 82
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 82
8009 7E MOV A,M 1
800A E6 ANI 8 bit 2
800B 01
800C C2 JNZ 16 bit 3
800D 80
800E 78
800F 78 MOV A,B 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
-MPS
85
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=00 B=00 C=00 D=78 E=80 H=82 L=02 PC=801E 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
800E 78
800F 78 MOV A,B 1
8010 86 ADD M 1
8011 47 MOV B,A 1
8012 23 INX H 1
8013 0C INR C 1
8014 80 ADD B 1
8015 0D DCR C 1
8016 C2 JNZ 16 bit 3
8017 09
8018 80
8019 78 MOV A,B 1
801A 32 STA 16 bit 3
801B 0A
801C 80
801D FF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Experiment: 11

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

Code:

```
MVI B, 0A

MVI D, AD

MVI C, 00

LXI H, 8500 H

BACK: MOV A, M

CMP D

JNZ NEXT

INR C

NEXT: INX H

DCR B

JNZ BACK

MOV A, C

STA 8600 H

RST 5
```

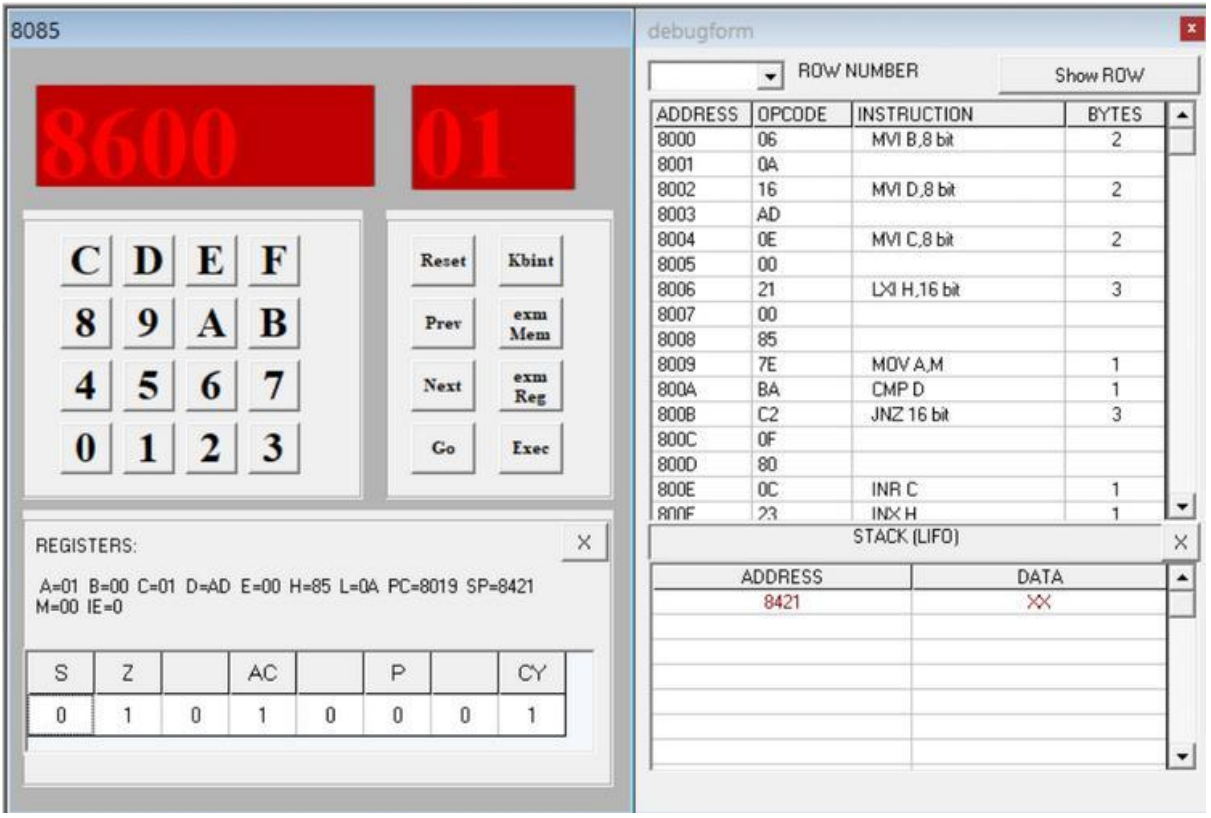
Assembly Language Code:

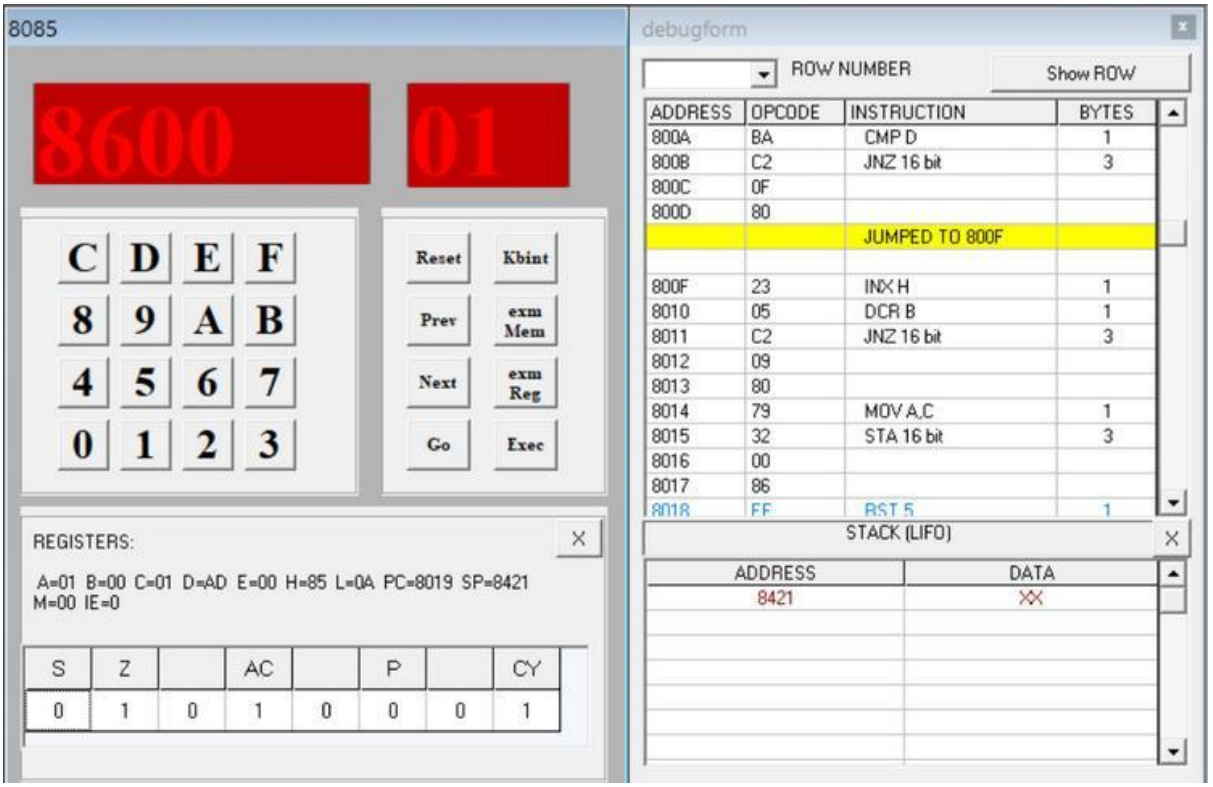
ADDRESS	INSTRUCTION	OPCODE
8000	MVI B,0A	06
8001		0A
8002	MVI D, AD	16
8003		AD
8004	MVI C, 00	0E
8005		00
8006	LXI H, 8500H	21
8007		00
8008		85
8009	BACK :MOV A, M	7E
800A	CMP D	BA
800B	JNZ NEXT	C2
800C		0F
800D		80
800E	INR C	0C
800F	NEXT: INX H	23
8010	DCR B	05
8011	JNZ BACK	C2
8012		09
8013		80
8014	MOV A,C	79
8015	STA 8600H	32
8016		00
8017		86
8018	RST 5	EF

INPUT [8500]=AD [8501]=8C [8502]=AD [8503]=34 [8504]=AD [8505]=56 [8506]=AD [8507]=AD [8508]=57 [8509]=AD

OUTPUT [8053]=06

Output:


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
800F 23 INX H 1
STACK (LIFO)
ADDRESS DATA
8421 XX


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
800A BA CMP D 1
800B C2 JNZ 16 bit 3
800C 0F
800D 80
JUMPED TO 800F
800F 23 INX H 1
8010 05 DCR B 1
8011 C2 JNZ 16 bit 3
8012 09
8013 80
8014 79 MOV A,C 1
8015 32 STA 16 bit 3
8016 00
8017 86
8018 FF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Experiment: 12

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

Code:

MVI B, 0A

MVI C, 00

LXI H, 8500H

BACK: MOV A, M

ANI FF

JPO NEXT

INR C

NEXT: INX H

DCR B

JNZ BACK

MOV A, C

STA 8600H

RST 5

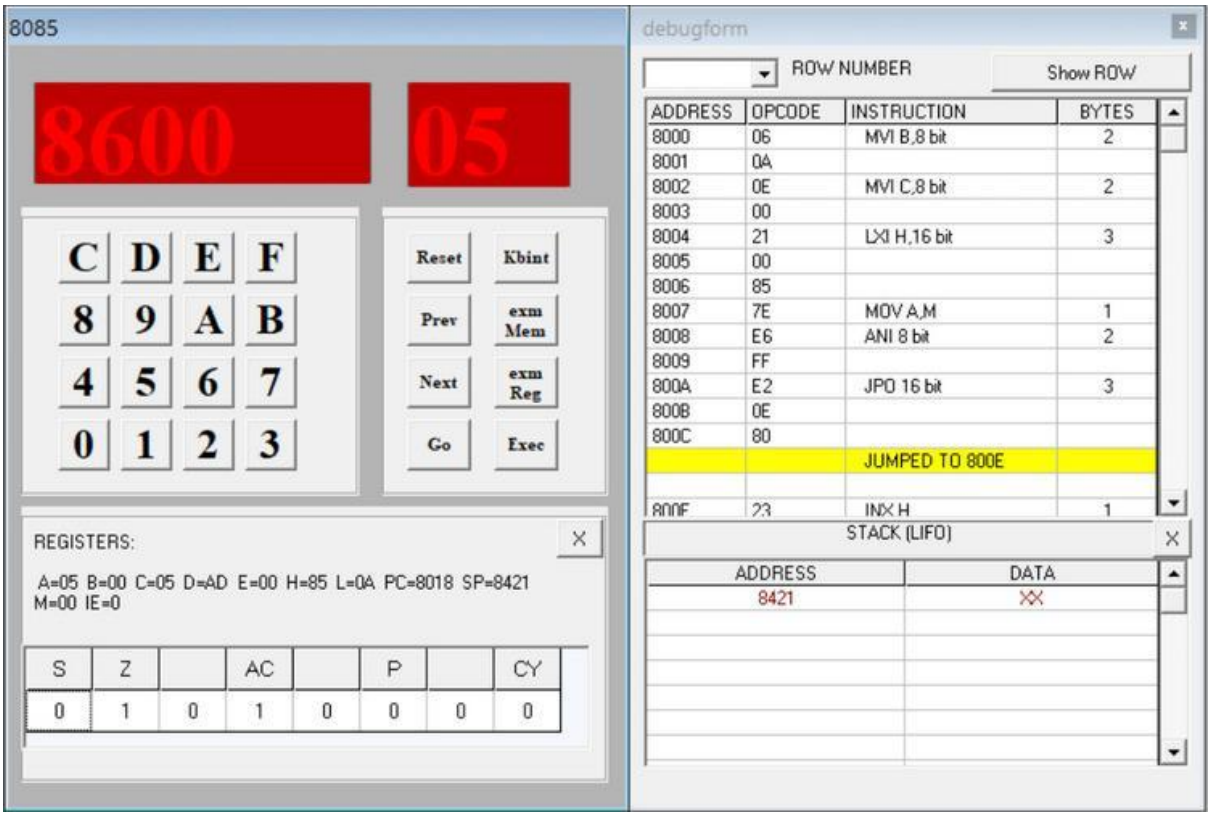
Assembly Language Code:

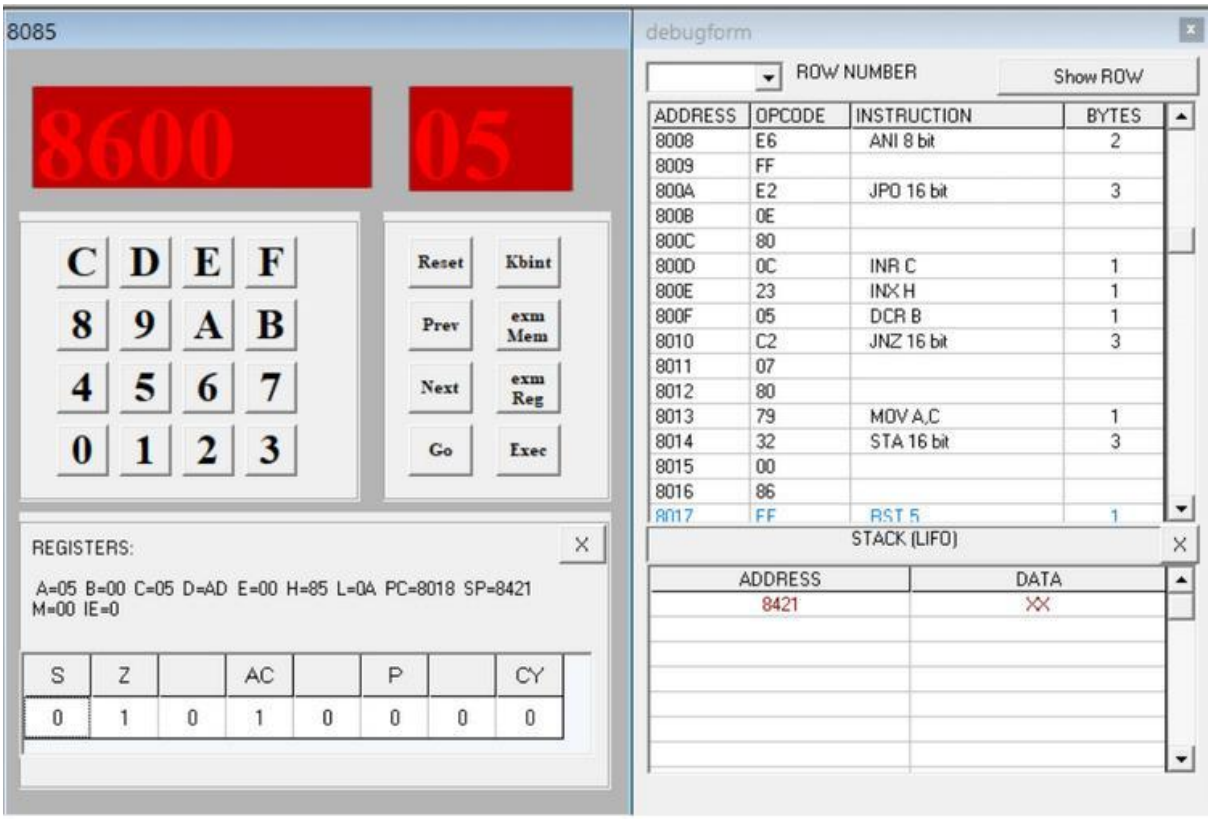
ADDRESS	INSTRUCTION	OPCODES
8000	MVI B, 0A	06
8001		0A
8002	MVI C, 00	0E
8003		00
8004	LXI H, 8500 H	21
8005		00
8006		85
8007	BACK: MOV A,M	7E
8008	ANI FF	E6
8009		FF
800A	JPO NEXT	E2
800B		0E
800C		80

800D	INR C	0C
800E	NEXT: INX H	23
800F	DCR B	05
8010	JNZ BACK	C2
8011		07
8012		80
8013	MOV A,C	79
8014	STA 8600 H	32
8015		00
8016		86
8017	RST5	EF

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

OUTPUT:




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=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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8008 E6 ANI 8 bit 2
8009 FF
800A E2 JPO 16 bit 3
800B 0E
800C 80
800D 0C INR C 1
800E 23 INX H 1
800F 05 DCR B 1
8010 C2 JNZ 16 bit 3
8011 07
8012 80
8013 79 MOV A,C 1
8014 32 STA 16 bit 3
8015 00
8016 86
8017 FF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Experiment: 13

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

Code:

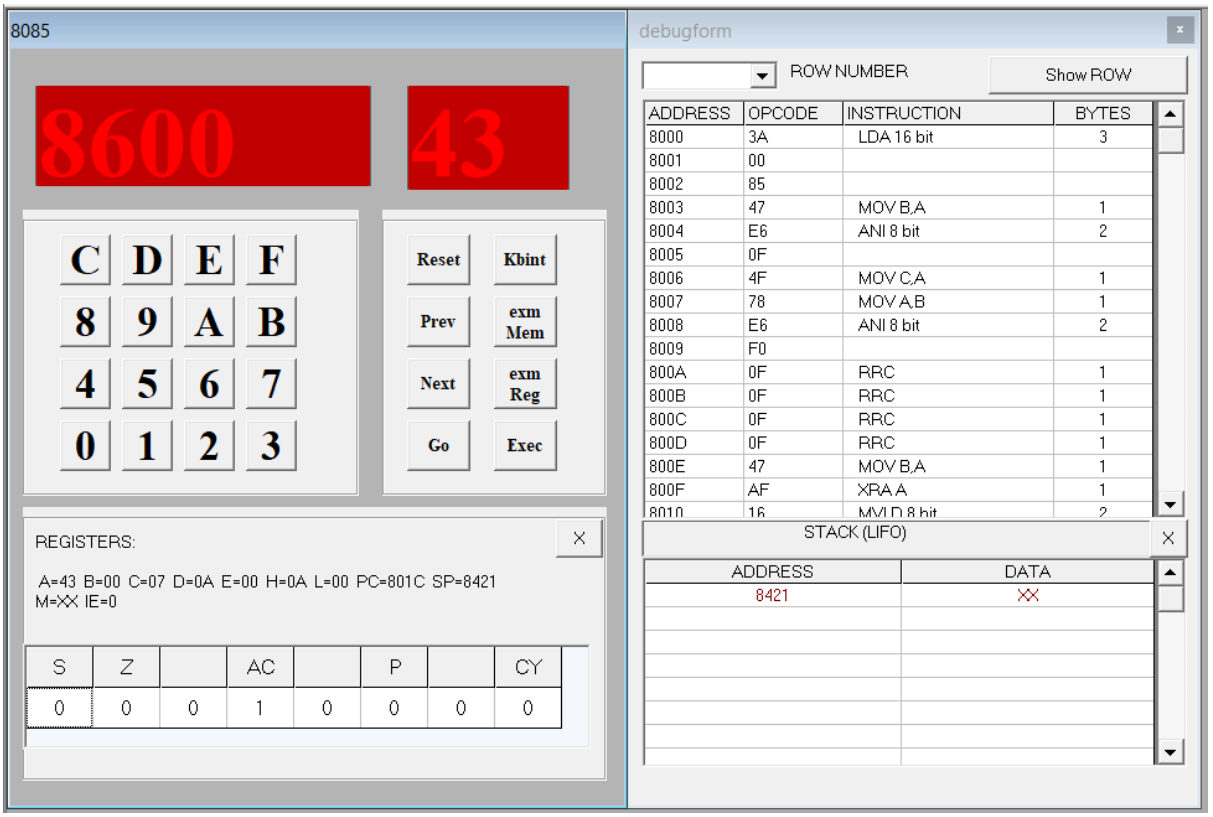
```
LDA 8500
MOV B,A
ANI 0F
MOV C,A
MOV A,B
ANI F0
RRC
RRC
RRC
RRC
MOV B,A
XRA A
MVI D, 0A
Sum : ADD D
DCR B
JNZ Sum
ADD C
STA 8600
RST 5
```

Assembly Language Code:

ADDRESS	INSTRUCTION	OPCODE
8000	LDA 8500	3A
8001		00
8002		85
8003	MOV B,A	47
8004	ANI 0F	E6
8005		0F
8006	MOV C,A	4F
8007	MOV A,B	78
8008	ANI F0	E6
8009		F0
800A	RRC	0F
800B	RRC	0F
800C	RRC	0F
800D	RRC	0F
800E	MOV B,A	47
800F	XRA A	AF
8010	MVI D, 0A	16
8011		0A
8012	Sum: ADD D	82
8013	DCR B	05
8014	JNZ Sum	C2
8015		12

8016		80
8017	ADD C	81
8018	STA 8600	32
8019		00
801A		86
801B	RST 5	EF

INPUT: [8500]=67
OUTPUT: [8600]=43



Experiment: 14

Objective: Write an assembly language program for exchanging the contents of memory location.

Code:

LDA 8500

MOV B,A

LDA 8600

STA 8500

MOV A, B

STA 8600

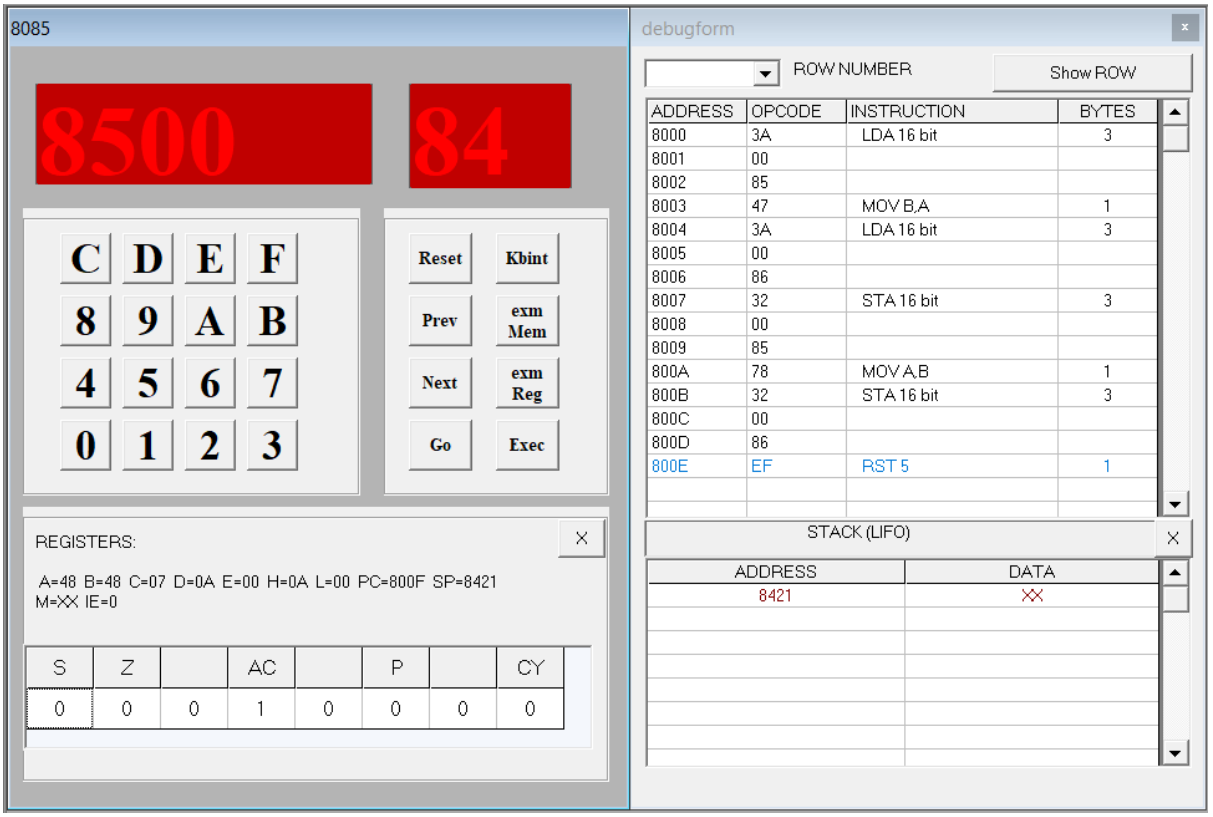
RST 5

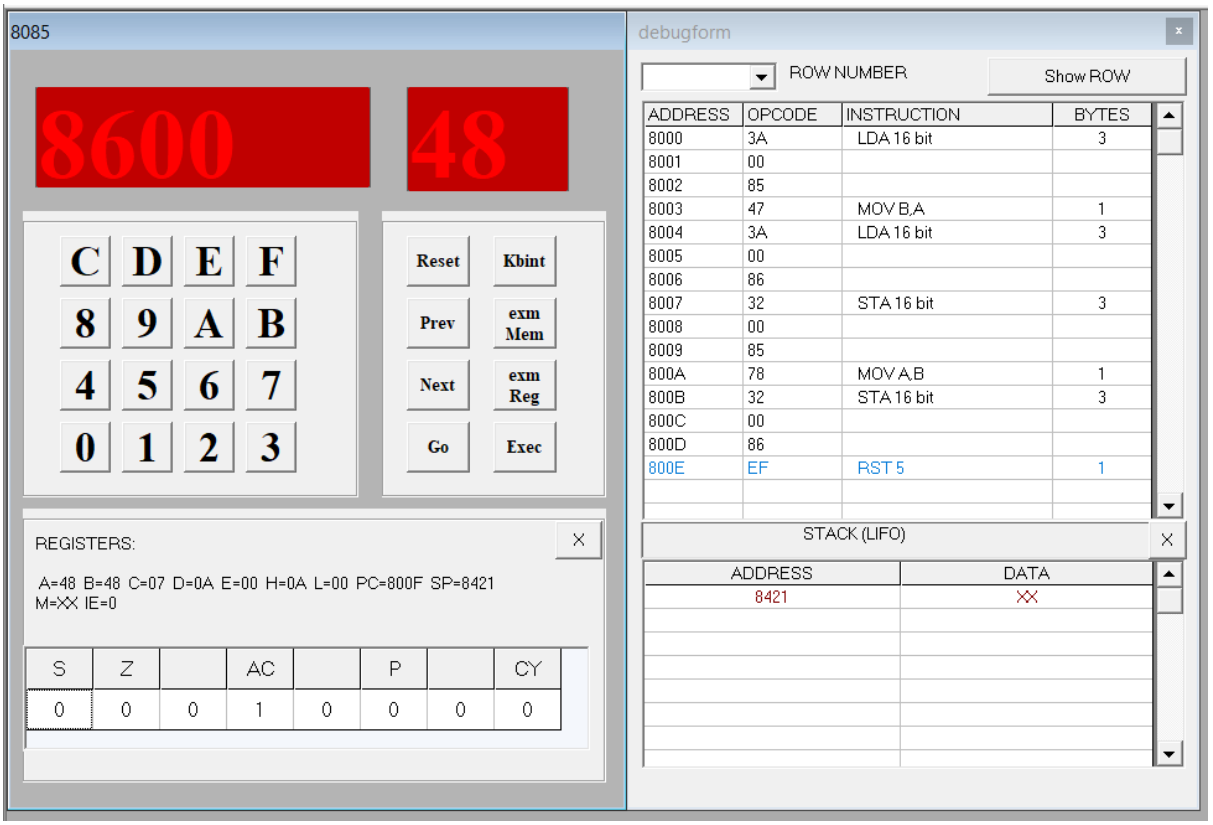
Assembly Language Code:

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

INPUT: [8500]=48 [8600]=84 **OUTPUT:** [8500]=84 [8600]=48

Output:


8085
8500
84
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=48 B=48 C=07 D=0A E=00 H=0A L=00 PC=800F SP=8421
M=XX 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 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


8085
8600
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=48 B=48 C=07 D=0A E=00 H=0A L=00 PC=800F SP=8421
M=XX 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 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

Experiment: 15

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

Code:

```
MVI B, 0A

LXI H,8500

 MOV A,M

Back: CMP M

JNC Next

MOV A,M

Next: INX H

DCR B

JNZ Back

STA 850A

RST 5
```

Assembly Language Program:

ADDRESS	INSTRUCTION	OPCODE
8000	MVI B, 0A	06
8001		0A
8002	LXI H,8500	21
8003		00
8004		85
8005	MOV A,M	7E
8006	Back: CMP M	BD
8007	JNC Next	D2
8008		09
8009		85
800A	MOV A,M	7E
800B	Next: INX H	23
800C	DCR B	05
800D	JNZ Back	C2
800E		06
800F		85
8010	STA 850A	32
8011		0A
8012		85
8013	RST 5	EF

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

OUTPUT: [850A]=0A