THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, PATIALA
MICROPROCESSOR-BASED SYSTEMS DESIGN

UCS617
2122EVESEM

Lab Assignment-1 (8085)

Submitted By:

Kunal Garg	101903683
Riya Mittal	101903687
Sameer Mahajan	101903686
Saksham Chauhan	101903682
Abhinav Goyal	101903684

Batch: 3COE26

Submitted To:

Dr. Manju Khurana
Assistant Professor, CSED
TIET, Patiala

INDEX

Problem 1:
Introduction of 8085-microprocessor kit and steps for execution on the kit.

Features of 8085 Microprocessor:

The Intel 8085 microprocessor is an NMOS 8-bit device. It has a 16-bit address bus and an 8- bit data bus. The total addressable memory size of 8085 microprocessor is 64 KB. It has a set of registers which contribute to the effective and efficient working of the microprocessor.

To view the overall working of the 8085 microprocessor, a kit has been designed so that the programming on this microprocessor can be best understood by the students.

The kit consists of the following components:

• A 6-byte display screen which is further divided into two parts, one containing 4-byte

• displaying the address and the remaining 2-bytes which are used to display the data.

• A keypad which is used to operate the kit.

• A 40-pin 8085 microprocessor.

• A 20-pin address latch used to manage the address transfer from the AD bus.

• A memory unit which consists of three 28-pin IC's which are used to provide memory to the processor.
• A 24-pin timer controller which is used to control the clock frequency.

Hardware Specifications:

• Operating System: Windows XP (sp3), Windows Vista, Windows 7 and above 32-bit and 64-bit operating systems.
• Minimum Screen Resolution: 1024x768
• Memory Space:3 to 5 MB
• .net Framework: 4.0.3210

Steps for Execution:

- 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

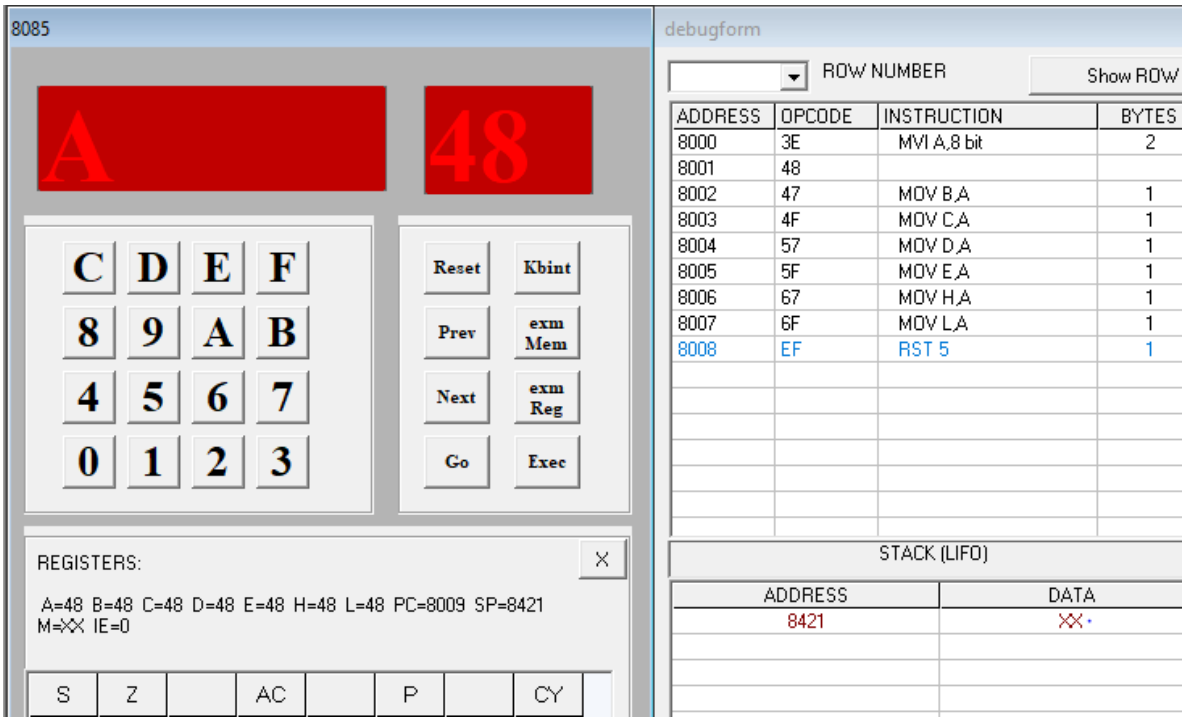
Program 2.1:
Write a program to store 8-bit data into one register and then copy that to all registers.

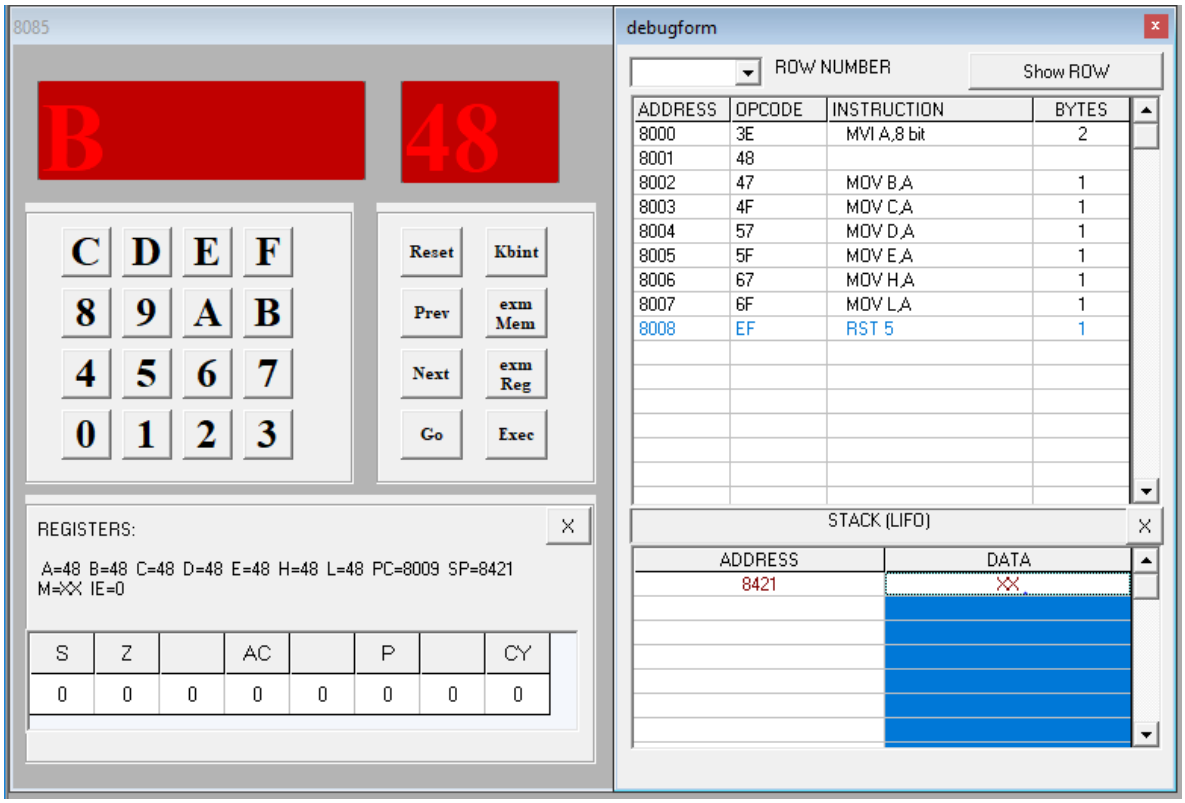
Code:

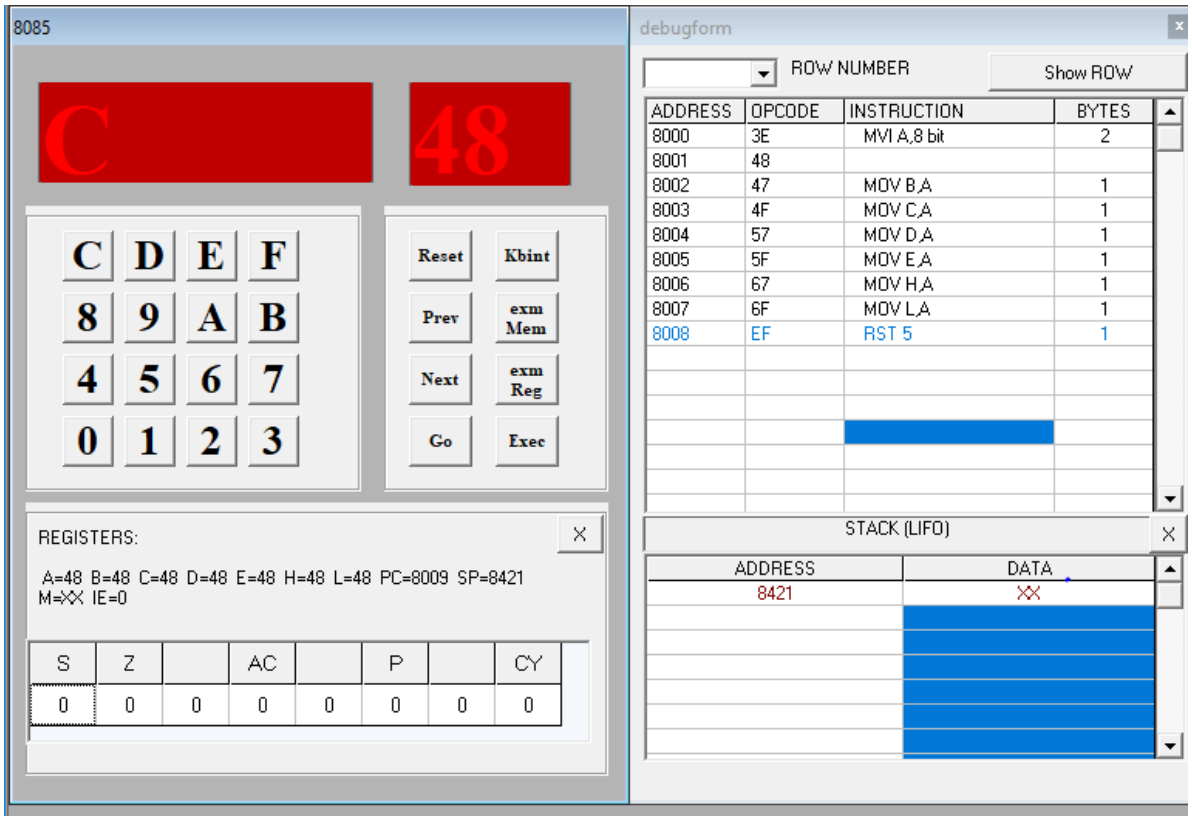
Code	**Memory Location**	**Opcode**
MVI A, 48	8000, 8001	3E, 48
MOV B, A	8002	47
MOV C, A	8003	4F
MOV D, A	8004	57
MOV E, A	8005	5F
MOV H, A	8006	67
MOV L, A	8007	6F
RST 5	8008	EF

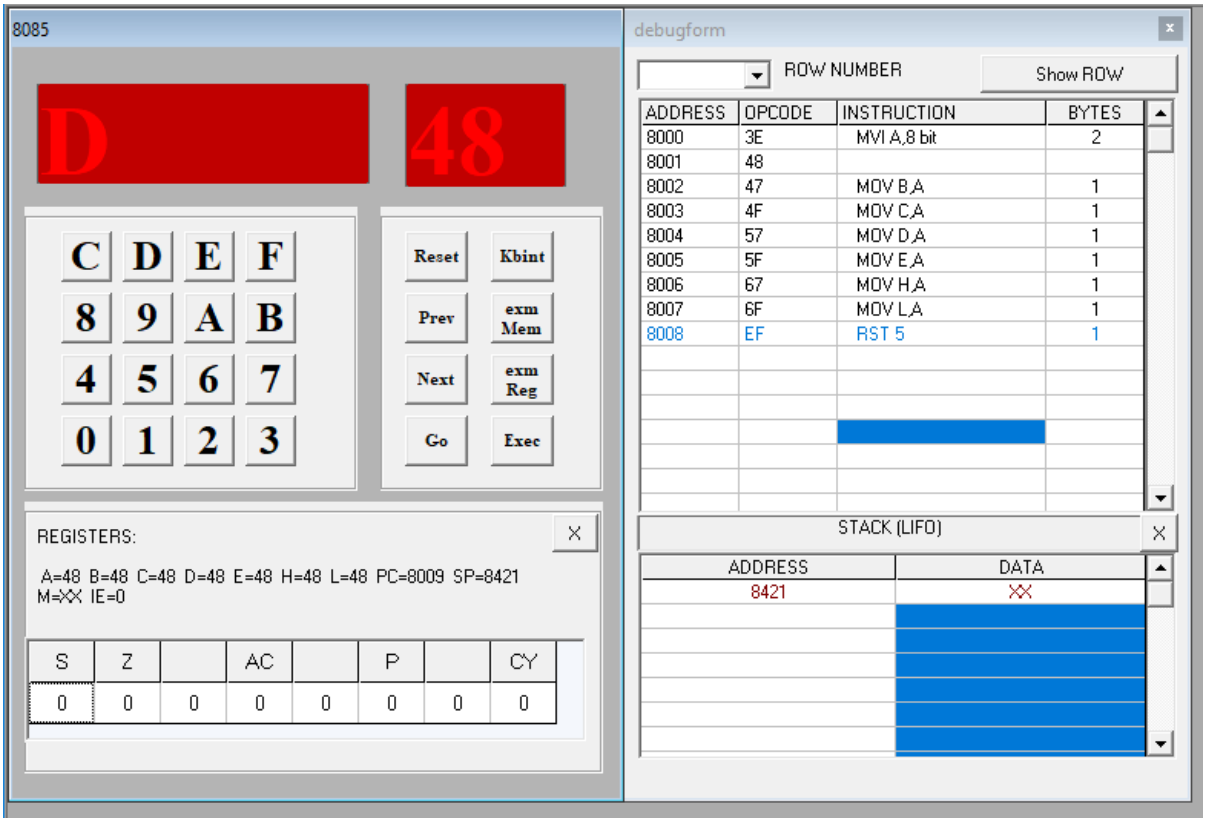
Output:
A – 48, B – 48, C – 48, D – 48, E – 48, H – 48, L – 48

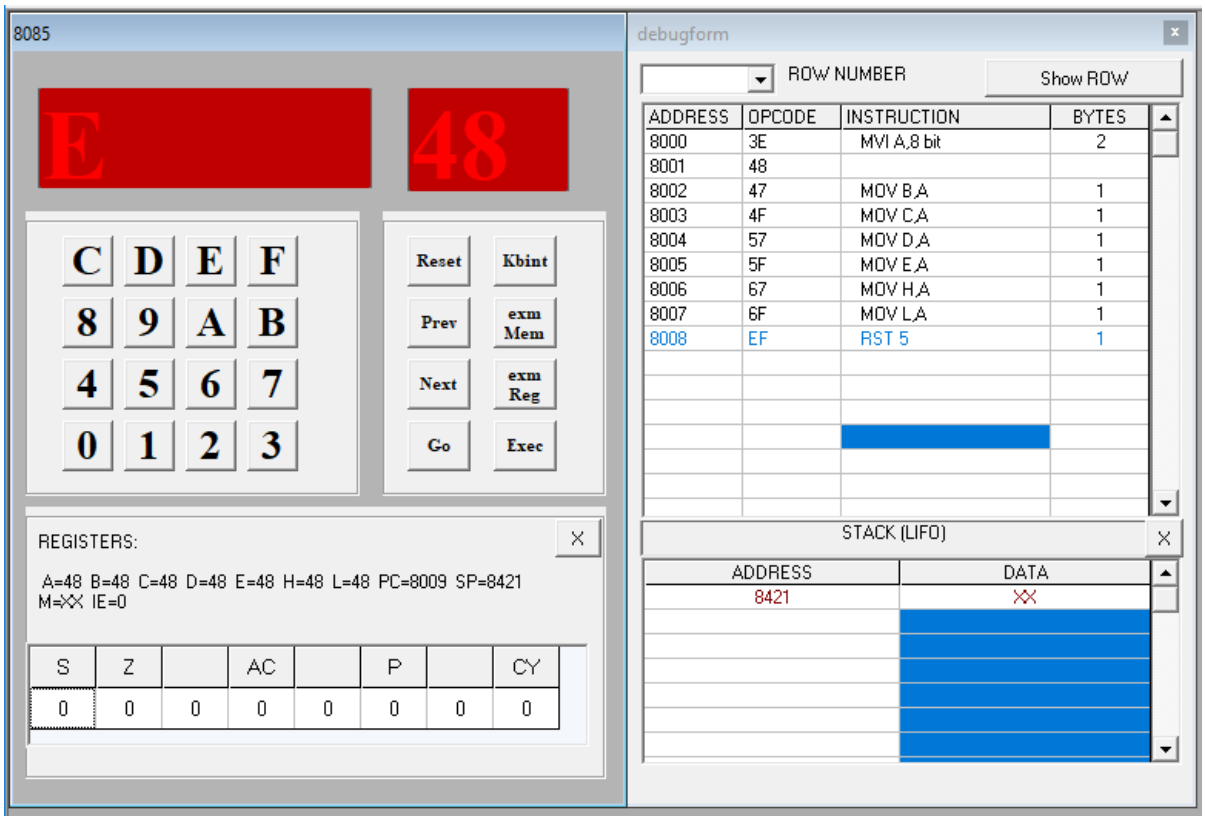
Simulation:

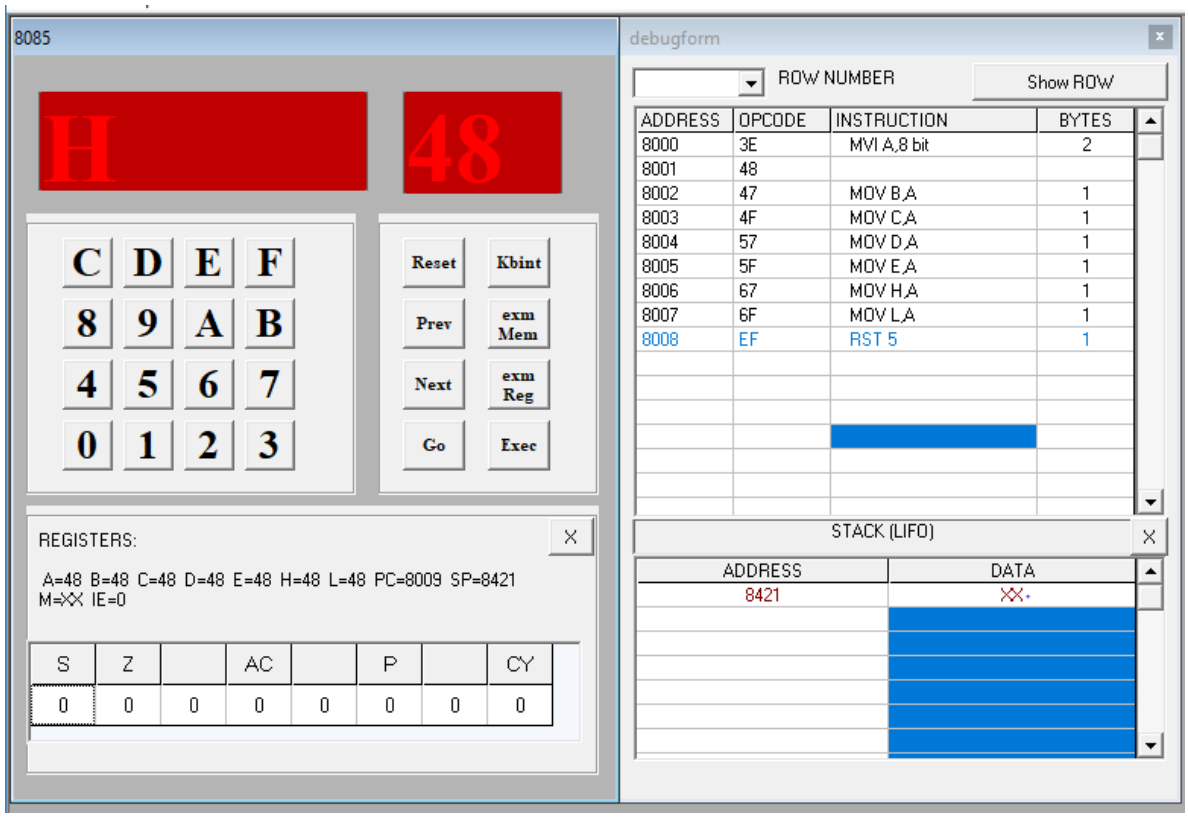

8085
A
48
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=48 B=48 C=48 D=48 E=48 H=48 L=48 PC=8009 SP=8421
M=XX IE=0
S Z AC P CY
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3E MVI A,8 bit 2
8001 48
8002 47 MOV B,A 1
8003 4F MOV C,A 1
8004 57 MOV D,A 1
8005 5F MOV E,A 1
8006 67 MOV H,A 1
8007 6F MOV L,A 1
8008 EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX

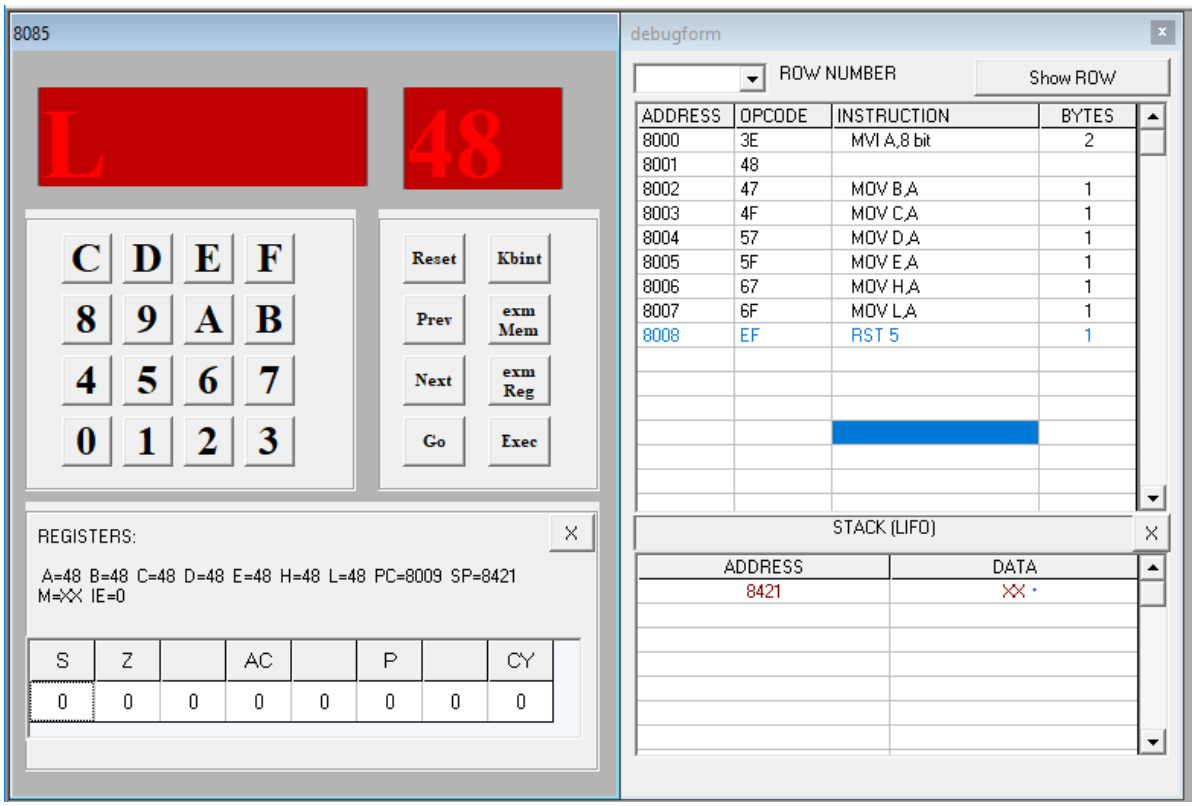

8085
B
48
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=48 B=48 C=48 D=48 E=48 H=48 L=48 PC=8009 SP=8421
M=XX IE=0
S Z AC P CY
0 0 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3E MVI A,8 bit 2
8001 48
8002 47 MOV B,A 1
8003 4F MOV C,A 1
8004 57 MOV D,A 1
8005 5F MOV E,A 1
8006 67 MOV H,A 1
8007 6F MOV L,A 1
8008 EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
C
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=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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3E MVI A,8 bit 2
8001 48
8002 47 MOV B,A 1
8003 4F MOV C,A 1
8004 57 MOV D,A 1
8005 5F MOV E,A 1
8006 67 MOV H,A 1
8007 6F MOV L,A 1
8008 EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
D
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=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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3E MVI A,8 bit 2
8001 48
8002 47 MOV B,A 1
8003 4F MOV C,A 1
8004 57 MOV D,A 1
8005 5F MOV E,A 1
8006 67 MOV H,A 1
8007 6F MOV L,A 1
8008 EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
E
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=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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3E MVI A,8 bit 2
8001 48
8002 47 MOV B,A 1
8003 4F MOV C,A 1
8004 57 MOV D,A 1
8005 5F MOV E,A 1
8006 67 MOV H,A 1
8007 6F MOV L,A 1
8008 EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
H
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=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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3E MVI A,8 bit 2
8001 48
8002 47 MOV B,A 1
8003 4F MOV C,A 1
8004 57 MOV D,A 1
8005 5F MOV E,A 1
8006 67 MOV H,A 1
8007 6F MOV L,A 1
8008 EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
L
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=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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3E MVI A,8 bit 2
8001 48
8002 47 MOV B,A 1
8003 4F MOV C,A 1
8004 57 MOV D,A 1
8005 5F MOV E,A 1
8006 67 MOV H,A 1
8007 6F MOV L,A 1
8008 EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX

Program 2.2:

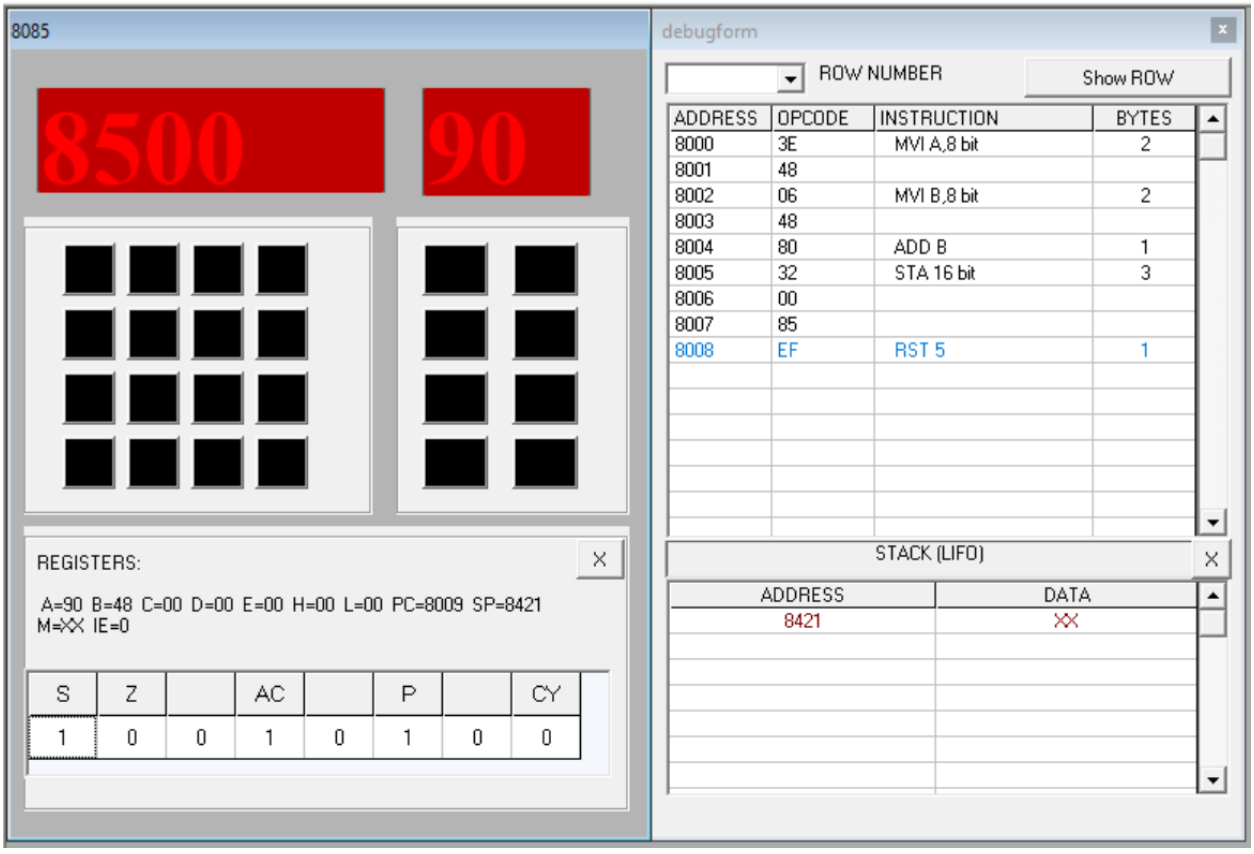
Write a program for addition of two 8-bit numbers.

Code:

Code	Memory Location	Opcode
MVI A, 48	8000, 8001	3E, 48
MOV B, 48	8002,8003	06,48
ADD B	8004	80
STA 8500	8005,8006,8007	32,00,85
RST 5	8008	EF

Output: [8500] - 90

Simulation:



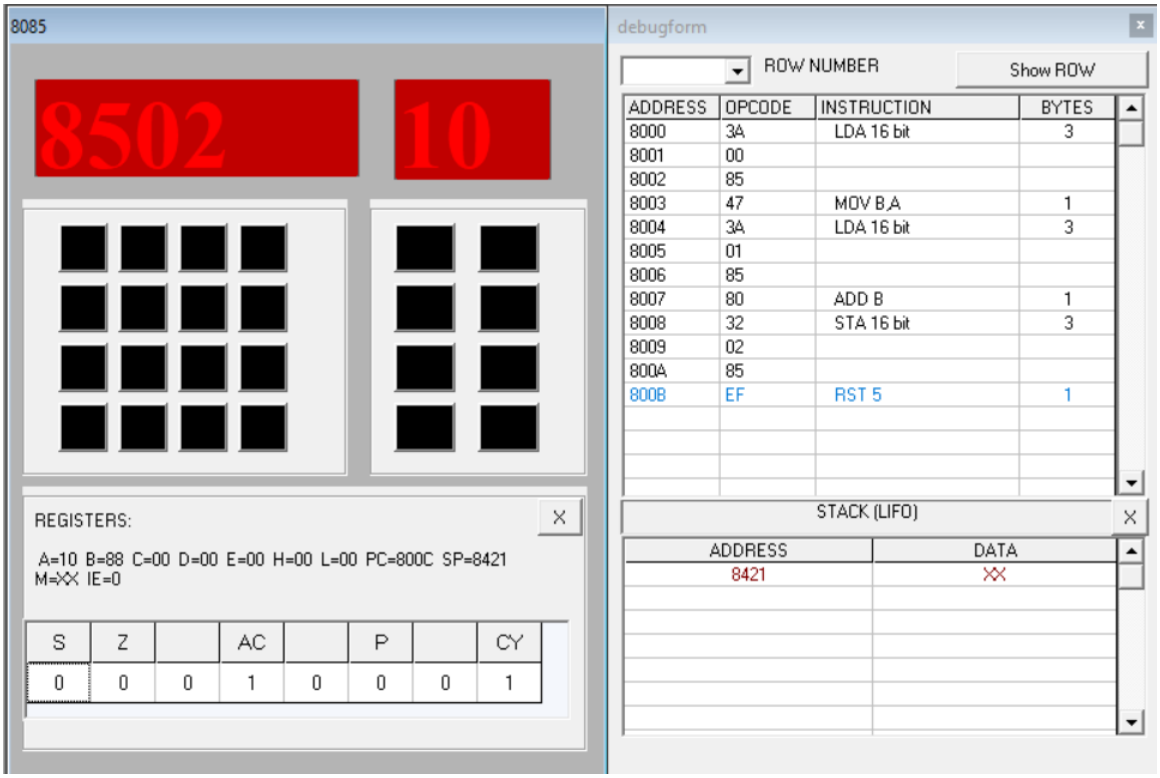
Program 2.3
Write a program to add 8-bit numbers using Direct addressing mode.

Code:

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

Input – [8500] - 88, [8501] - 88
Output – [8502] - 10

Simulation:

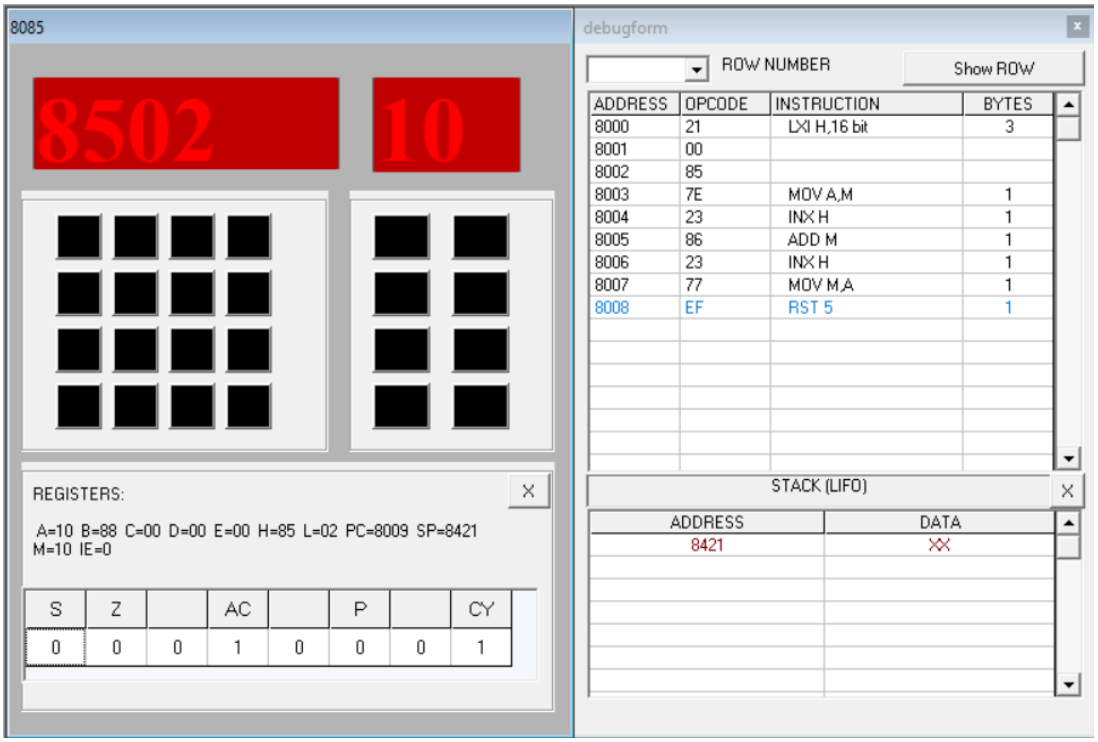


Program 2.3:
Write a program to add 8-bit numbers using Indirect addressing mode.

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

Input – [8500] - 88, [8501] - 88
Output – [8502] – 10

Simulation:



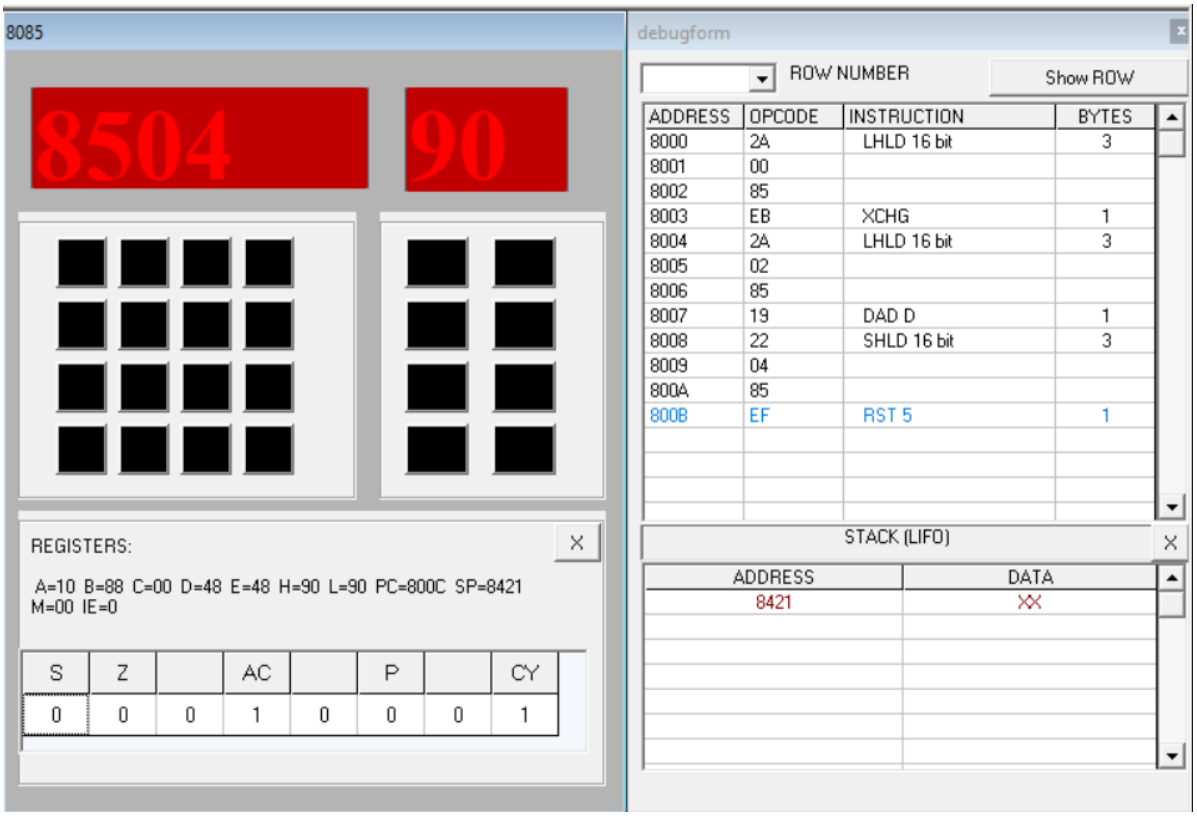
Program 2.4
Write a program to add 16-bit numbers using Direct addressing mode.

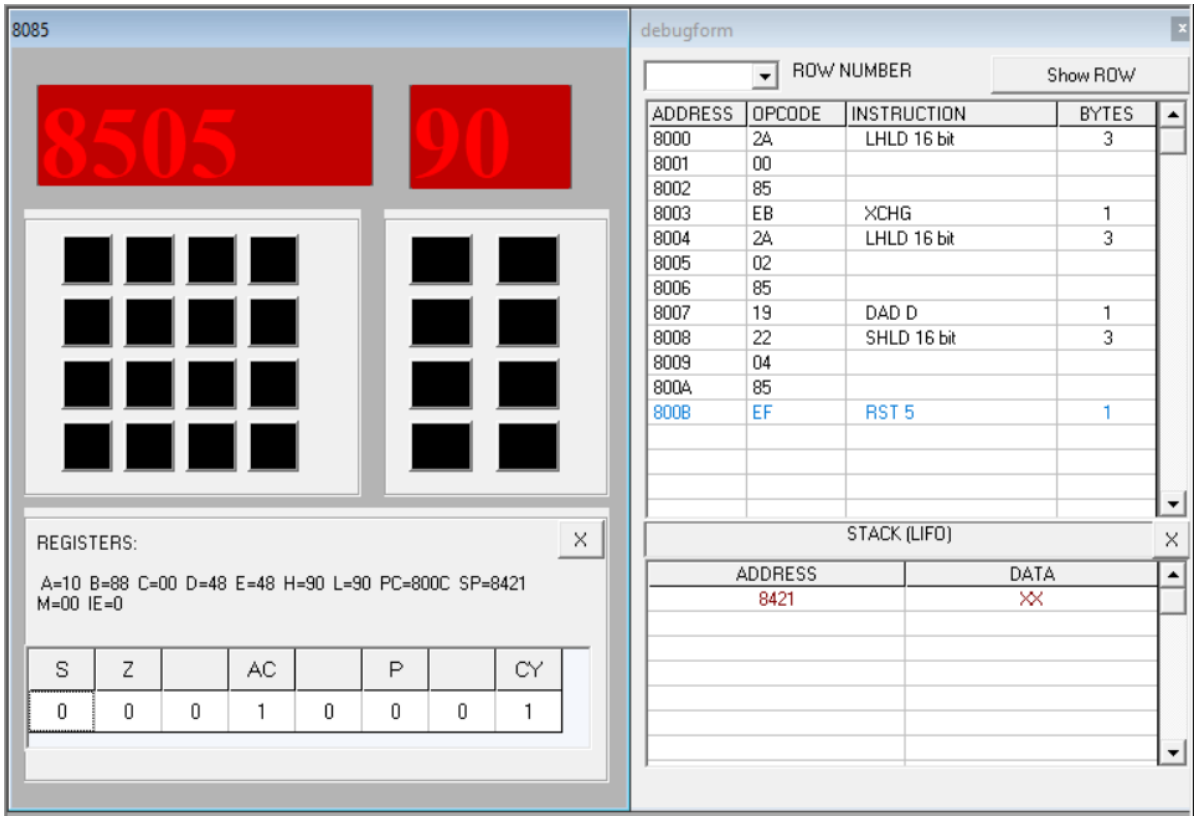
Code:

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

Input – [8500] - 48, [8501] - 48, [8502] - 48, [8503] - 48
Output – [8504] - 90, [8505] - 90

Simulation:


8085
8504
90
REGISTERS:
A=10 B=88 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 1 0 0 0 1
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
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
8505
90
REGISTERS:
A=10 B=88 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 1 0 0 0 1
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
STACK (LIFO)
ADDRESS DATA
8421 XX

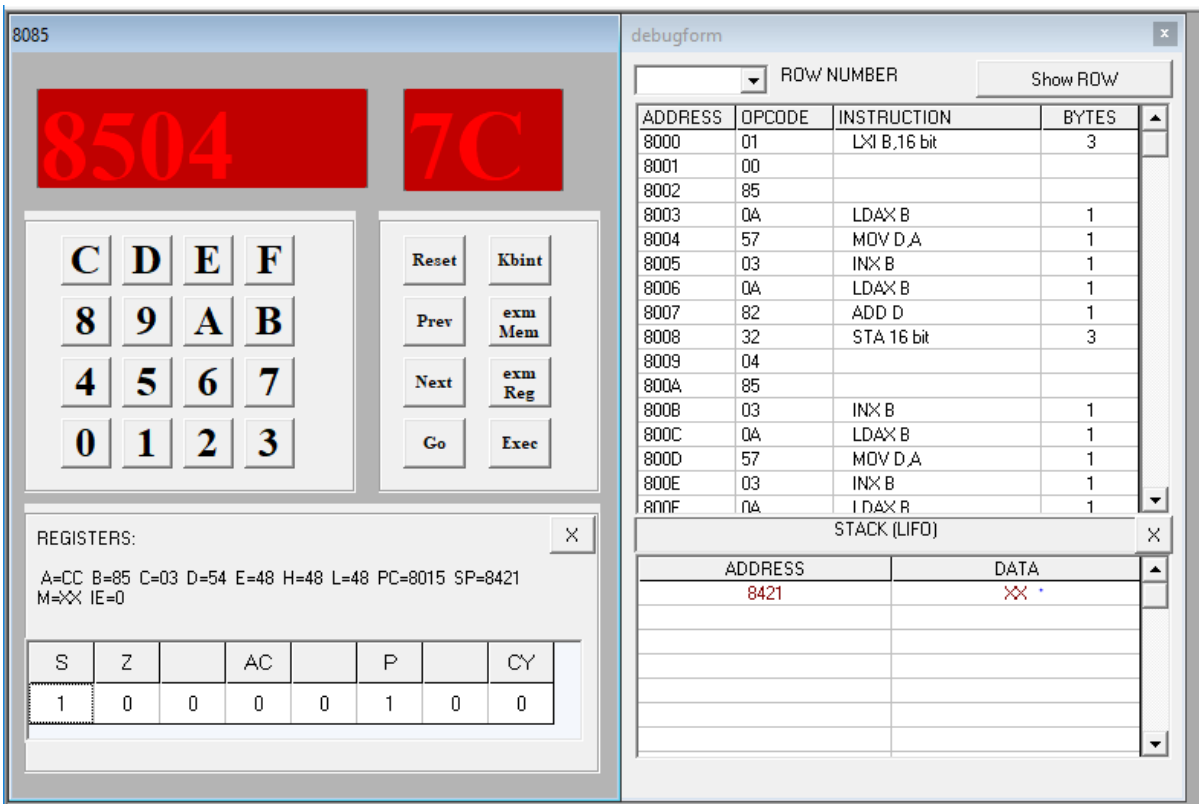
Program 2.4
Write a program to add 16-bit numbers using Indirect addressing mode.

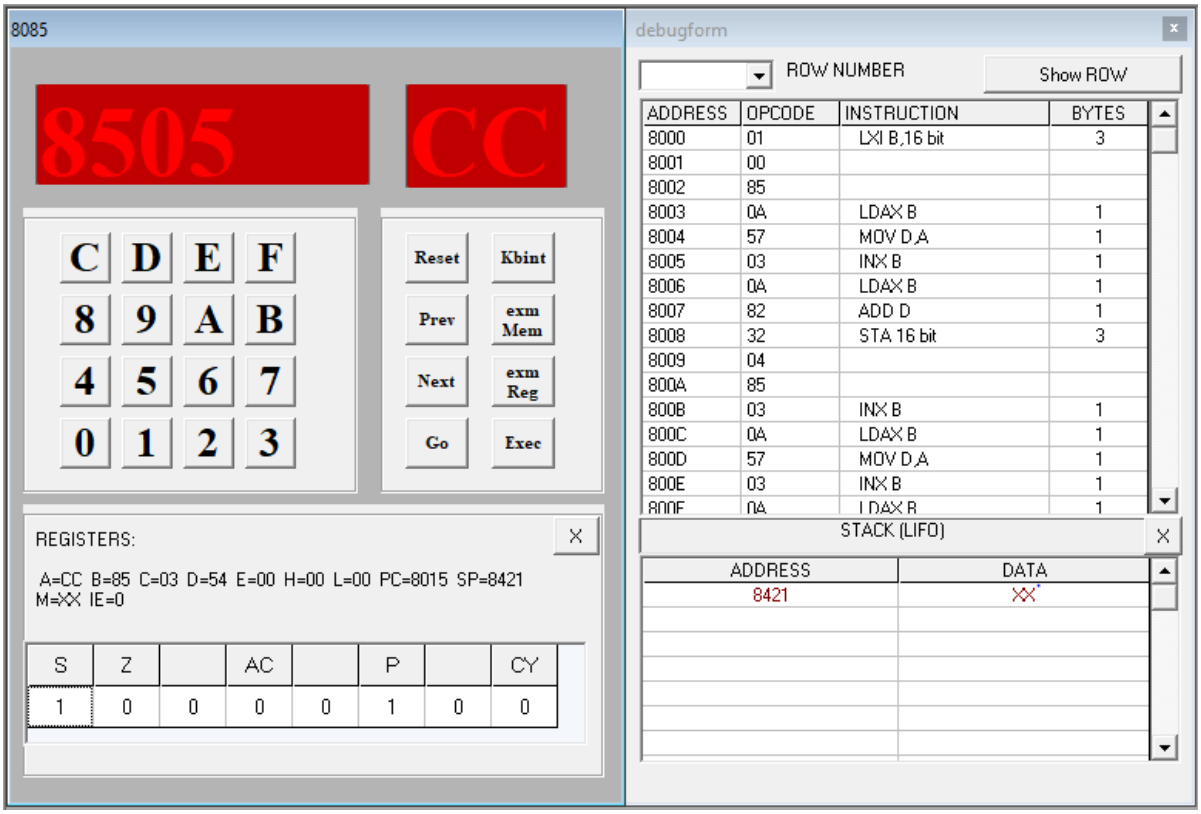
Code:

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	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

Input – [8500] - 34, [8501] - 48, [8502] - 54, [8503] - 78
Output – [8504] - 7C , [8505] - CC

Simulation:


8085
8504
7C
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=48 H=48 L=48 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

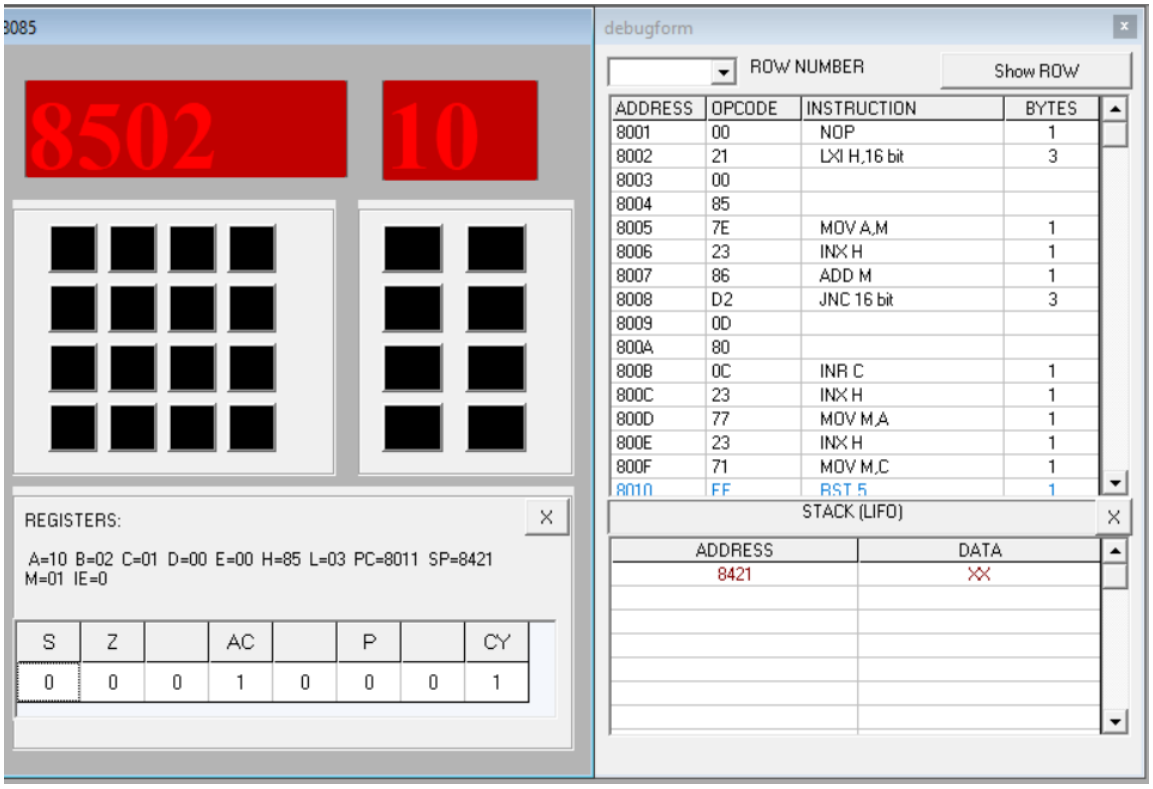

8085
8505
CC
C D E F
8 9 A B
4 5 6 7
0 1 2 3
Reset
Kbint
Prev
exm Mem
Next
exm Reg
Go
Exec
REGISTERS:
A=CC B=85 C=03 D=54 E=00 H=00 L=00 PC=8015 SP=8421
M=XX IE=0
S Z AC P CY
1 0 0 0 0 1 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 01 LXI B,16 bit 3
8001 00
8002 85
8003 0A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

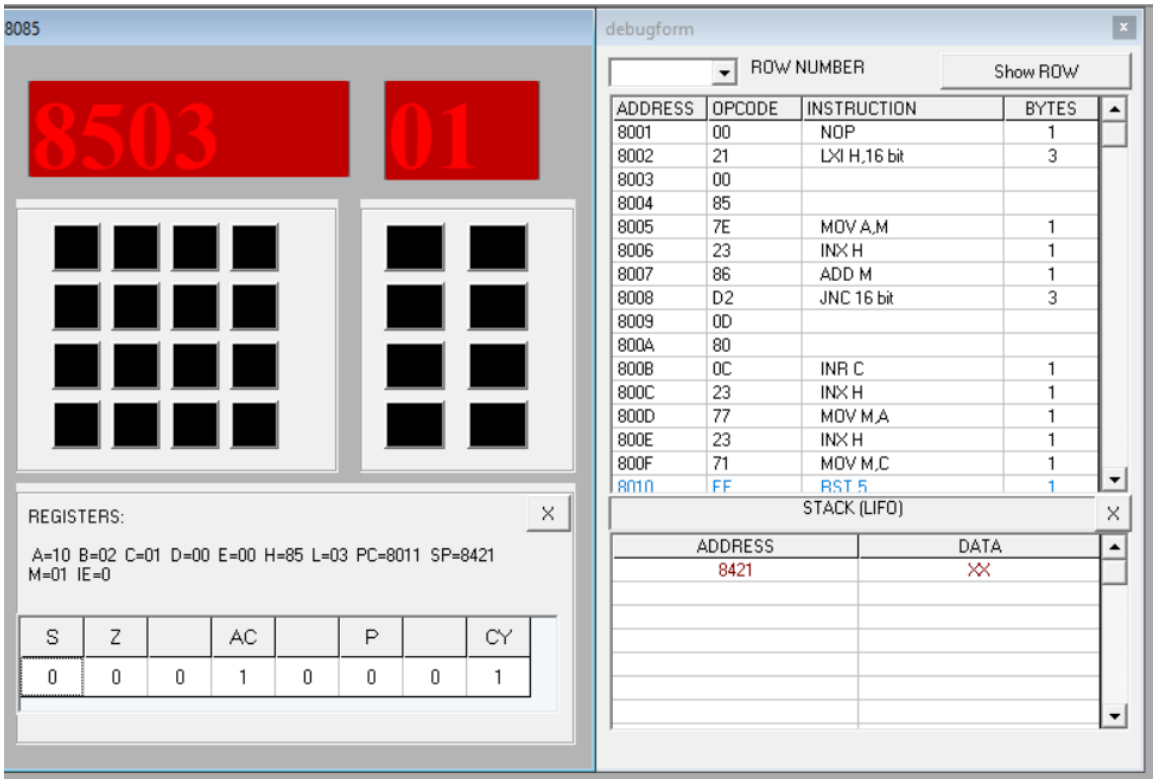
Program 2.5:
Write a program to 8-bit numbers using carry (using JNC instruction).

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, 0D, 80
INR C	800B	0C
INX H	800C	23
Next: MOV M, A	800D	77
INX H	800E	23
MOV M, C	800F	71
RST 5	8010	EF

Input - [8500] – 88, [8501] – 88
Output - [8502] – 10, [8503] – 01

Simulation:


8085
8502
10
REGISTERS:
A=10 B=02 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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8001 00 NOP 1
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 0D
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 FF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
8503
01
REGISTERS:
A=10 B=02 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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8001 00 NOP 1
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 0D
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 FF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX

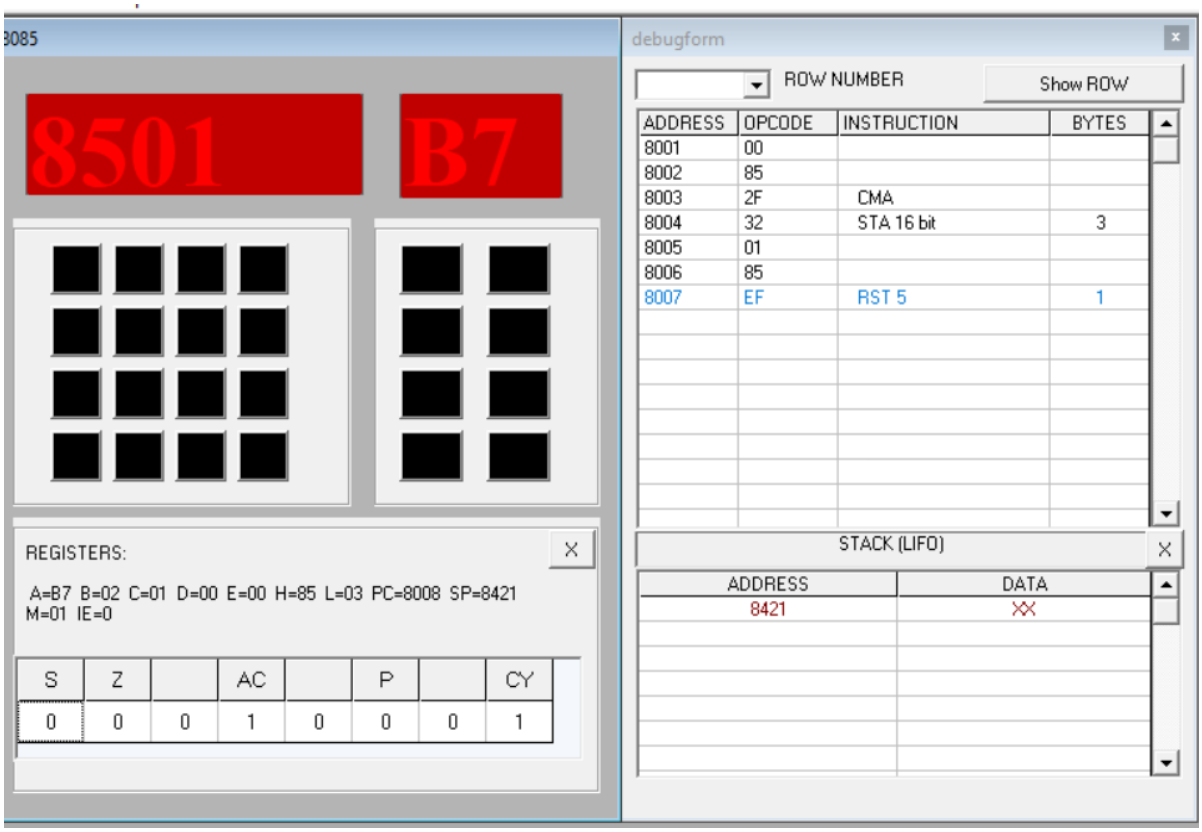
Program 2.6:

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

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

Input - [8500] – 48

Output - [8501] – B7

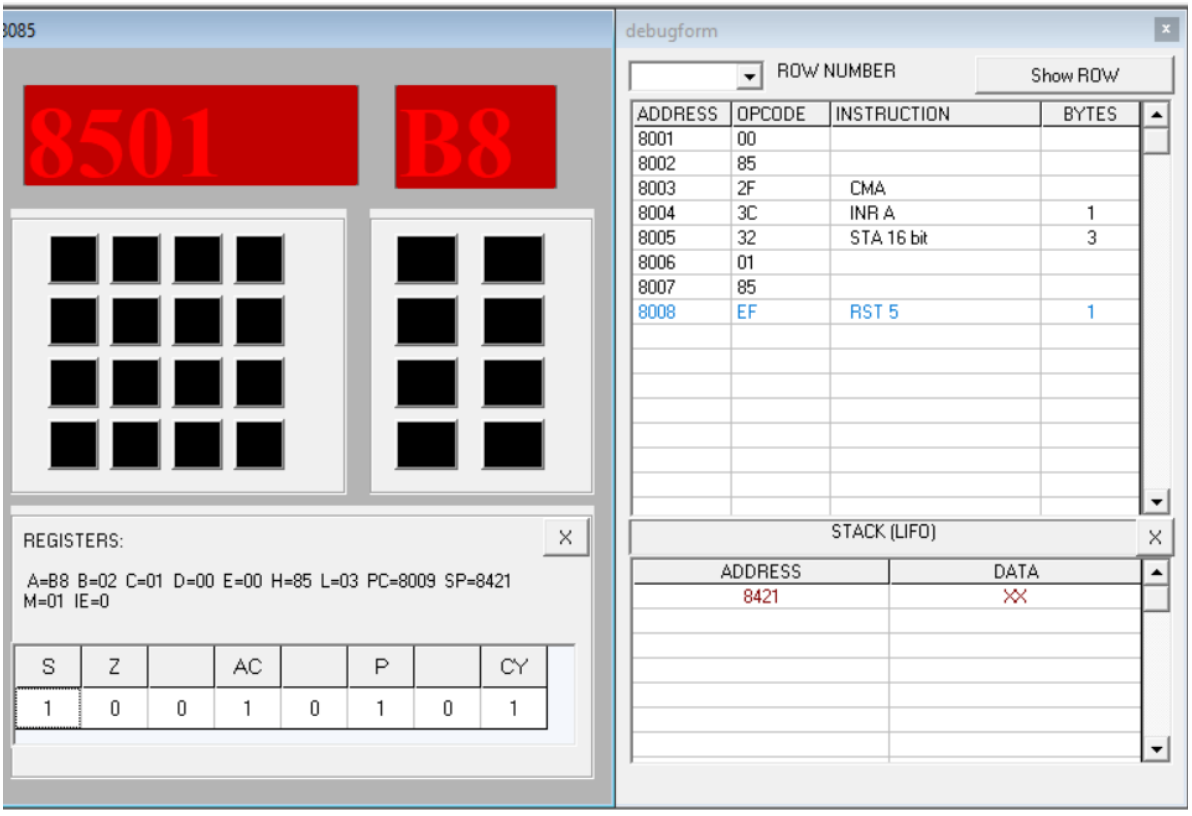


Problem : 2's complement of 8-bit numbers

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

Input - [8500] – 48
Output - [8501] – B8

Simulation:



Program 3:

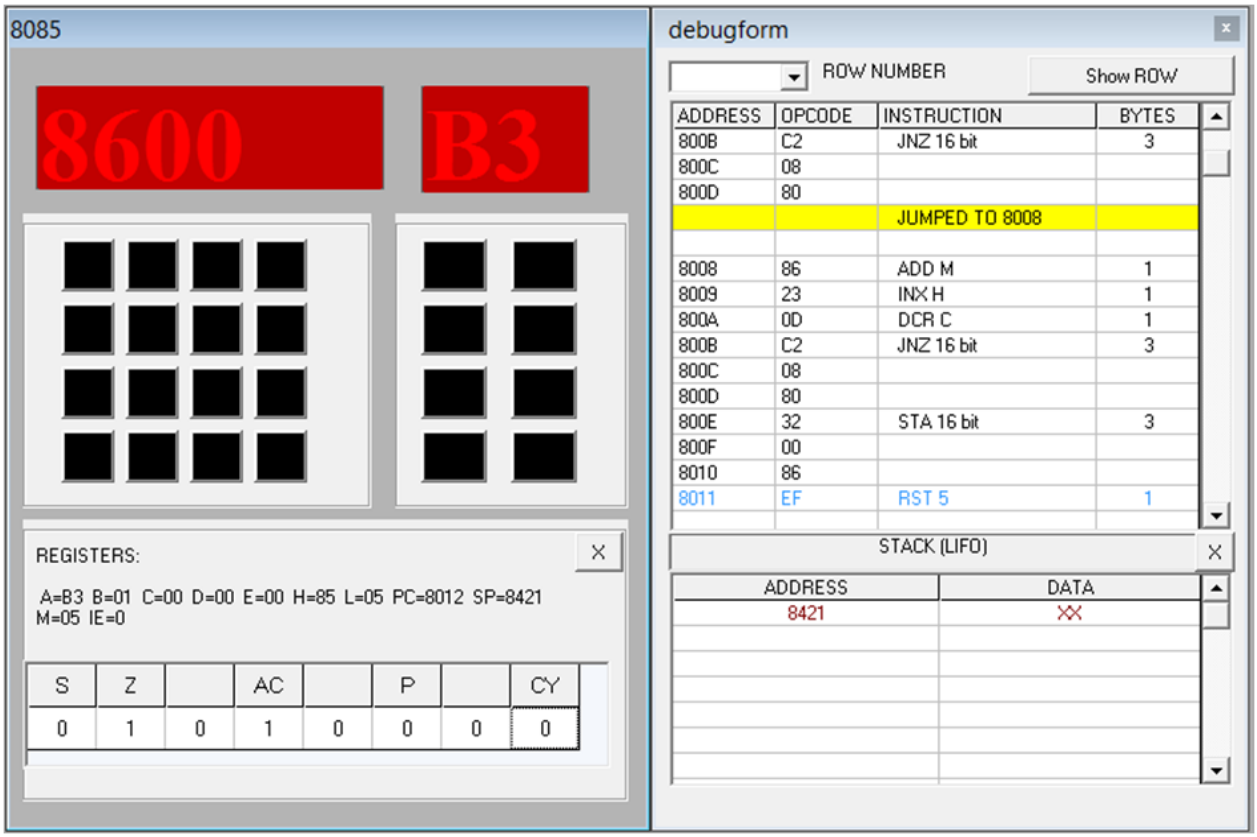
Write a program for the sum of a series of numbers.

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

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

Result - 1B3

Output - [8600] – B3


8085
8600
B3
REGISTERS:
A=B3 B=01 C=00 D=00 E=00 H=85 L=05 PC=8012 SP=8421
M=05 IE=0
S Z AC P CY
0 1 0 1 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
800B C2 JNZ 16 bit 3
800C 08
800D 80
JUMPED TO 8008
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
800E 32 STA 16 bit 3
800F 00
8010 86
8011 EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX

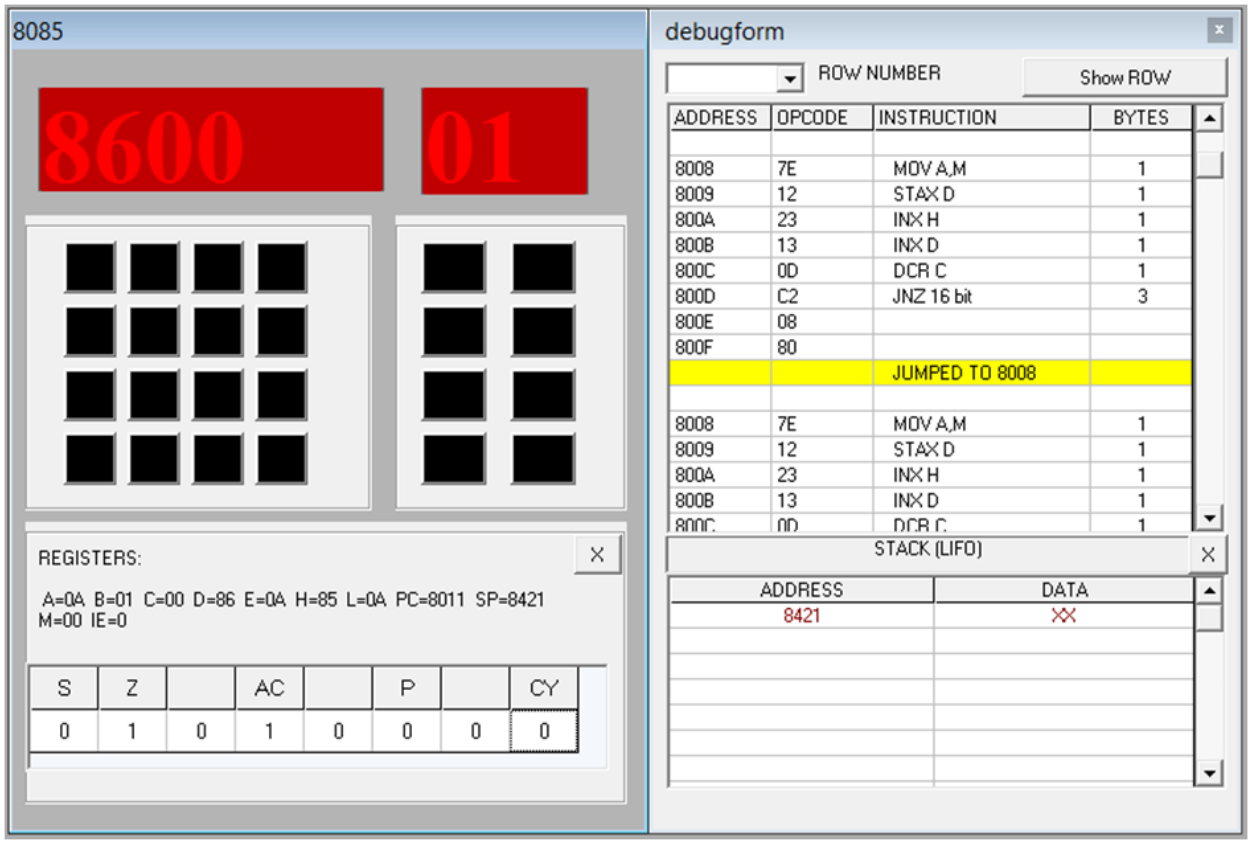
Program 4:
Write a program for data transfer from memory block B1 to memory block

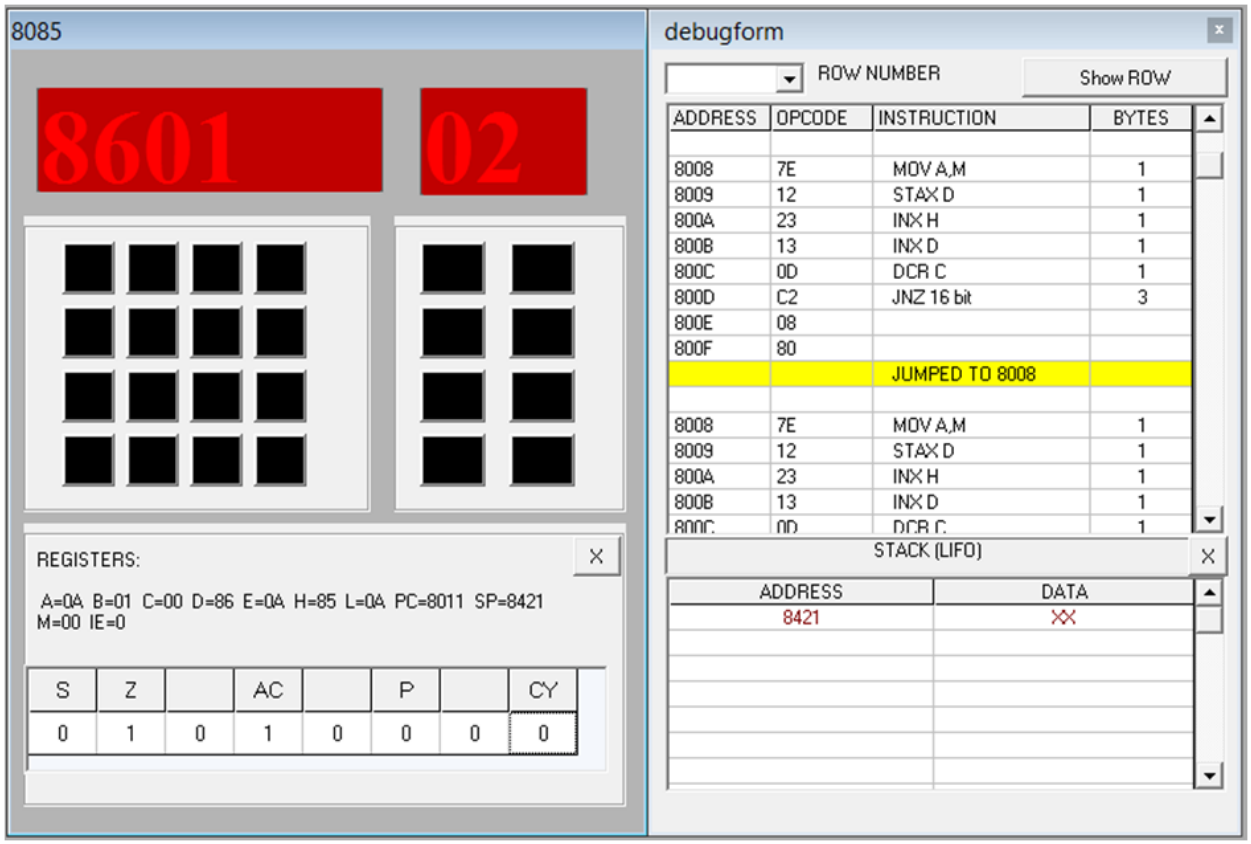
Code	Memory Location	Opcode
MVI C, 0AH	8000, 8001	0E,0A
LXI H, 8500H	8002, 8003, 8004	21,00,85
LXI D,8600H	8005, 8006, 8007	11,00,86
BACK: MOV A, M	8008	7E
STAX D	8009	12
INX H	800A	23
INX D	800B	13
DCR C	800C	0D
JNZ Back	800D,800E,800F	C2,08,80
RST 5	8010	EF

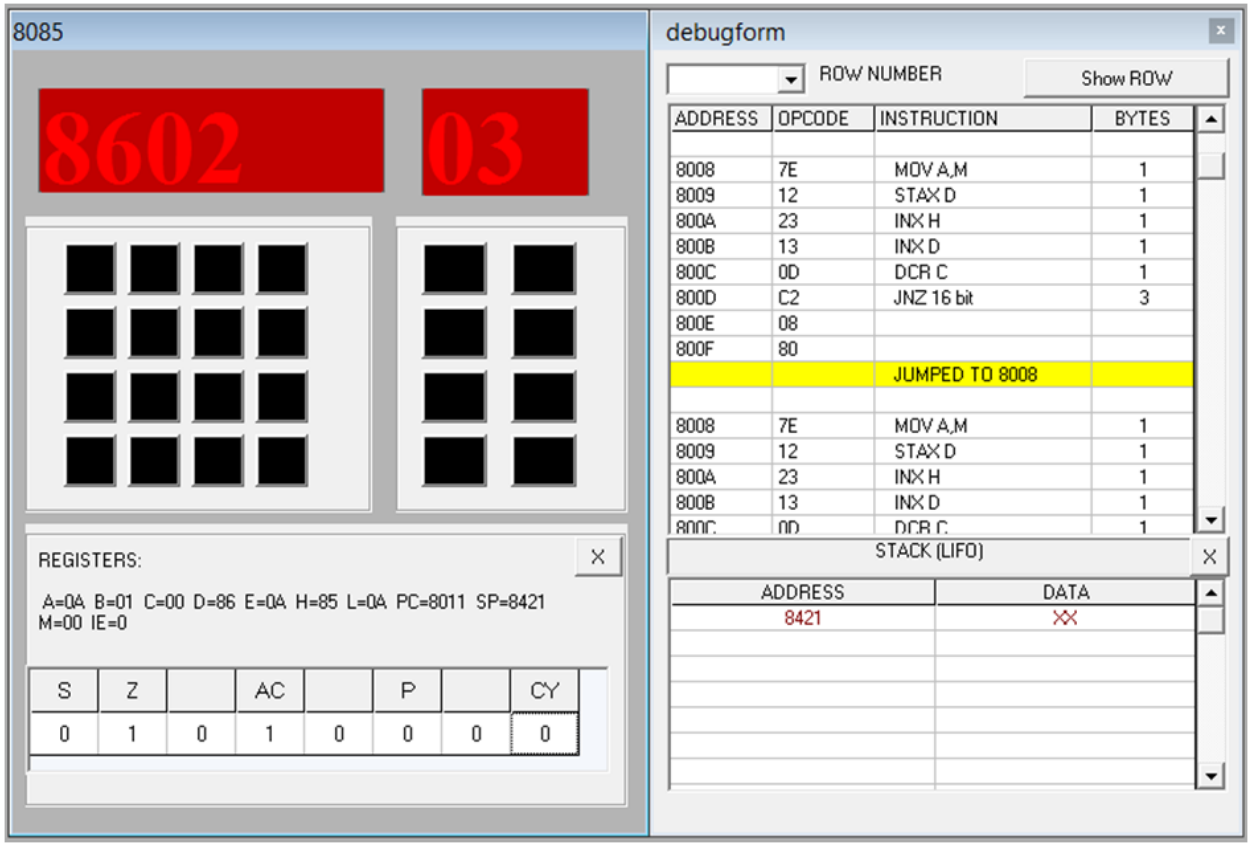
Input - [8500] – 01, [8501] – 02, [8502] – 03 … [8509] – 0A

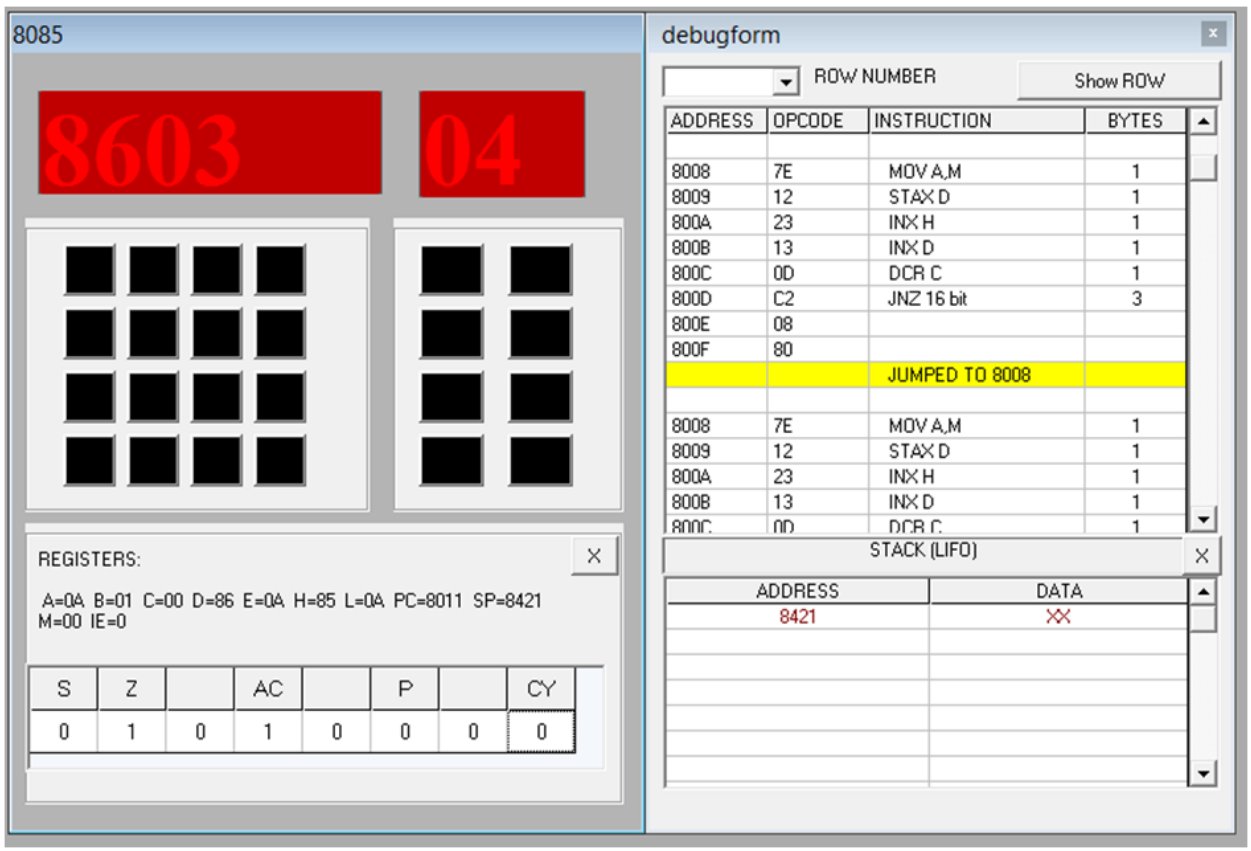
Output - [8600] – 01, [8601] – 02, [8602] – 03 … [8609] – 0A

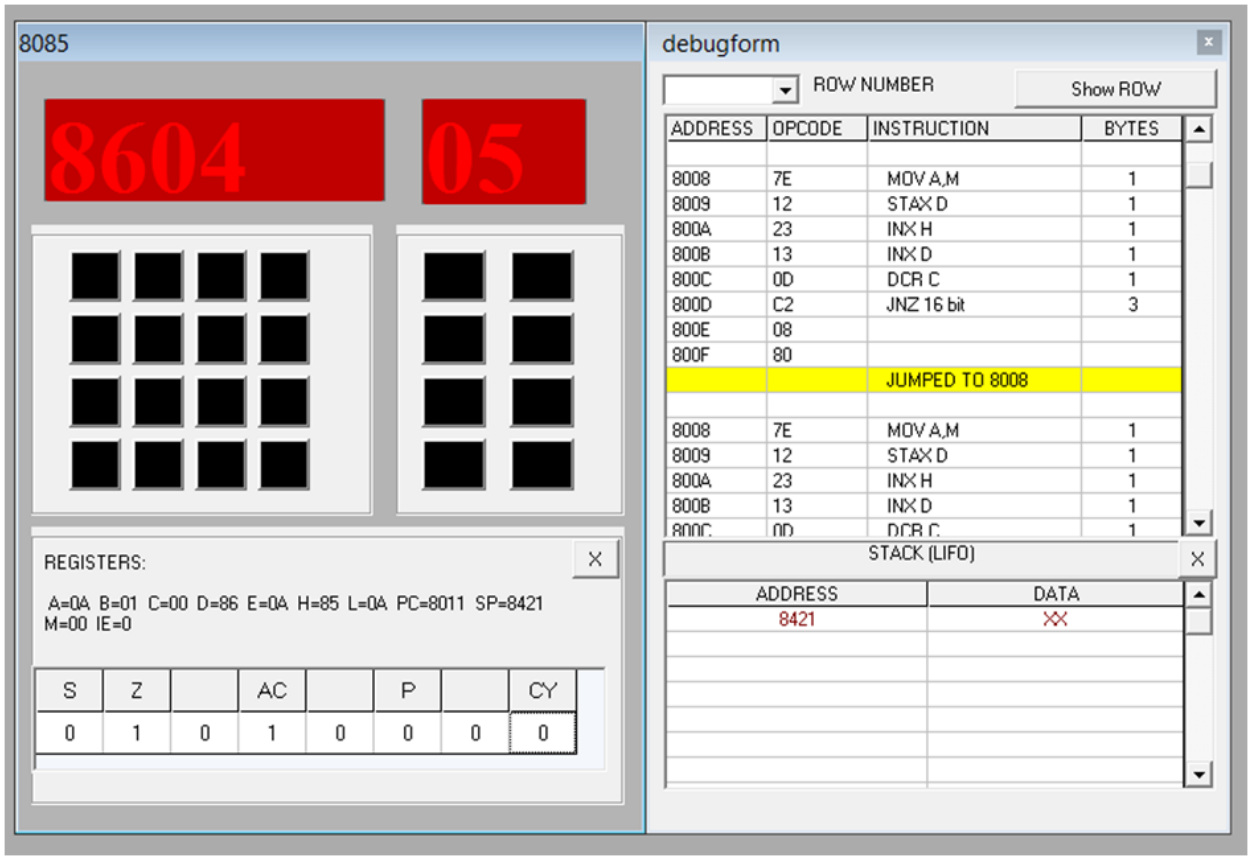
Simulation:

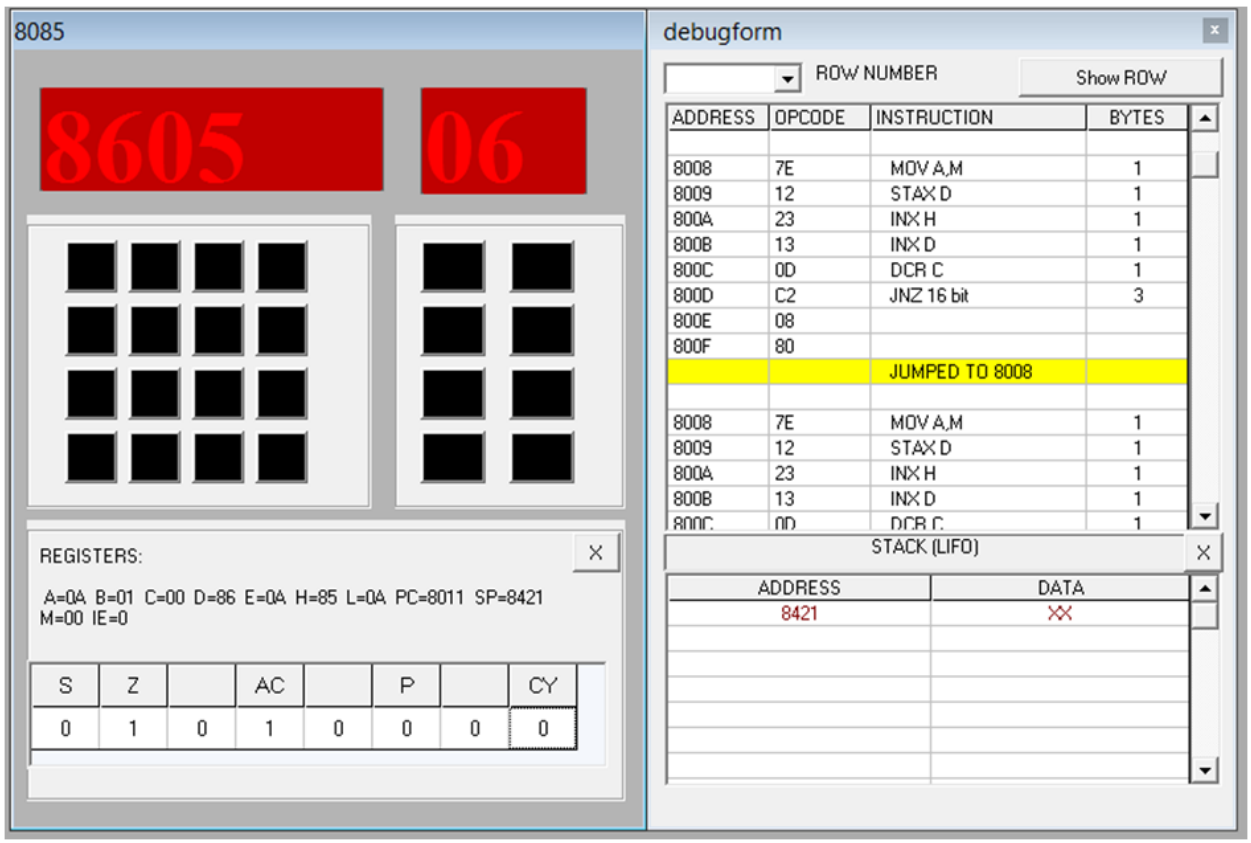

8085
8600
01
REGISTERS:
A=0A B=01 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 1 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
JUMPED TO 8008
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
STACK (LIFO)
ADDRESS DATA
8421 XX

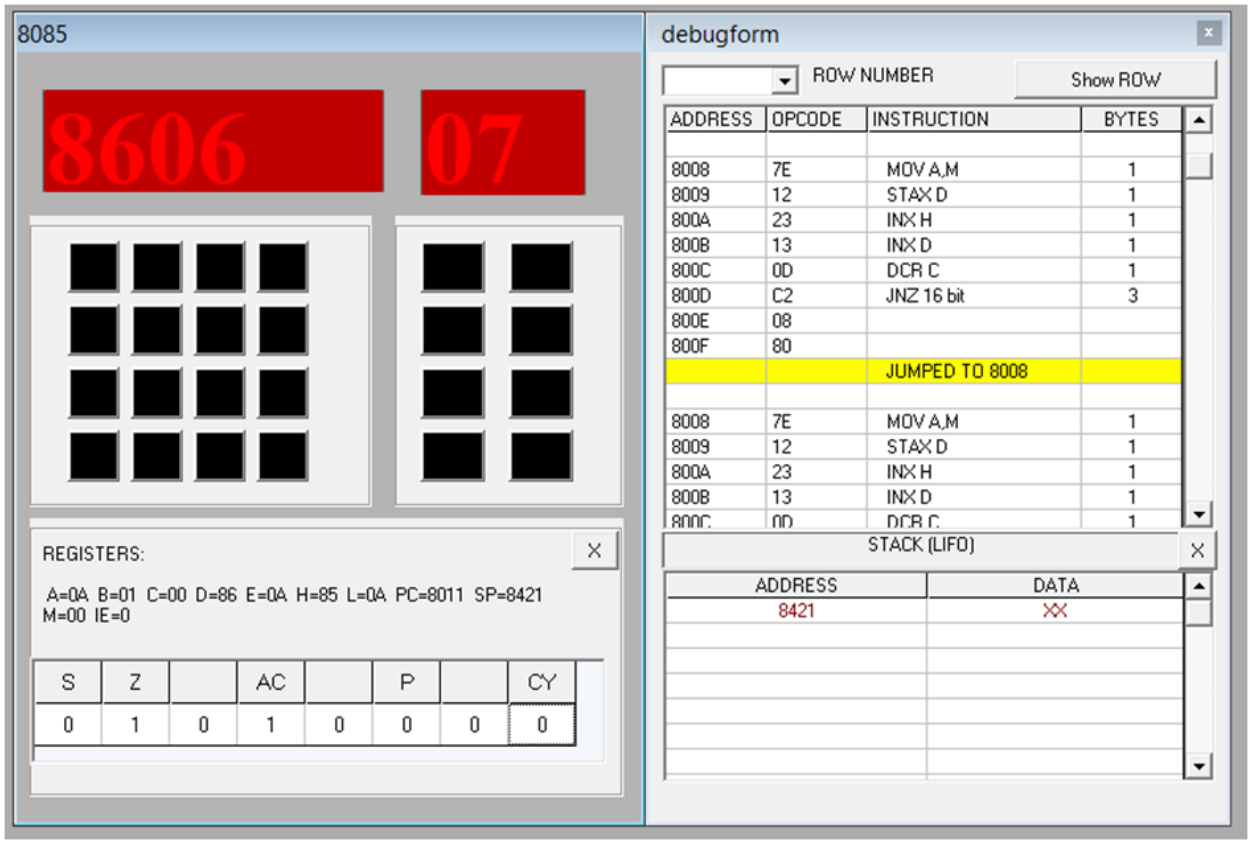

8085
8601
02
REGISTERS:
A=0A B=01 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 1 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
JUMPED TO 8008
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
STACK (LIFO)
ADDRESS DATA
8421 XX

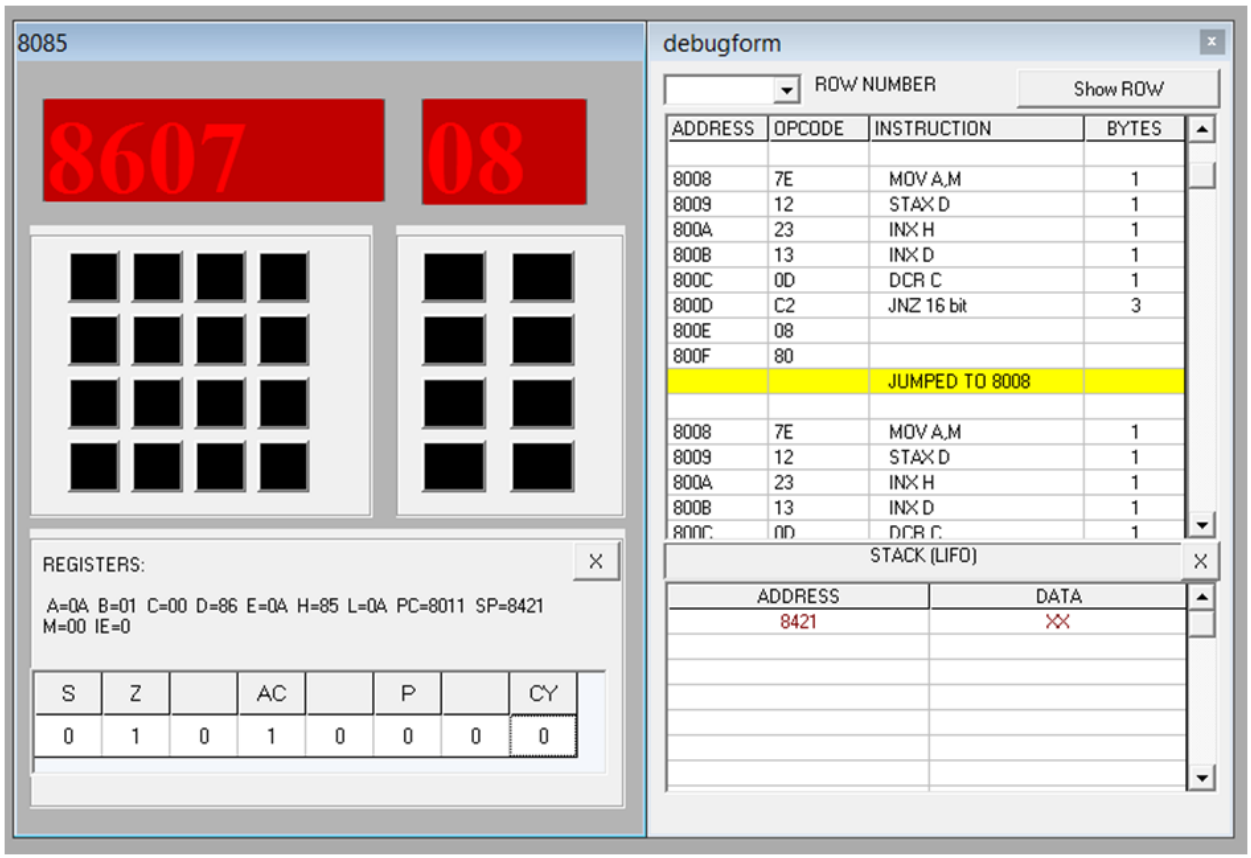

8085
8602
03
REGISTERS:
A=0A B=01 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 1 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
JUMPED TO 8008
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
STACK (LIFO)
ADDRESS DATA
8421 XX

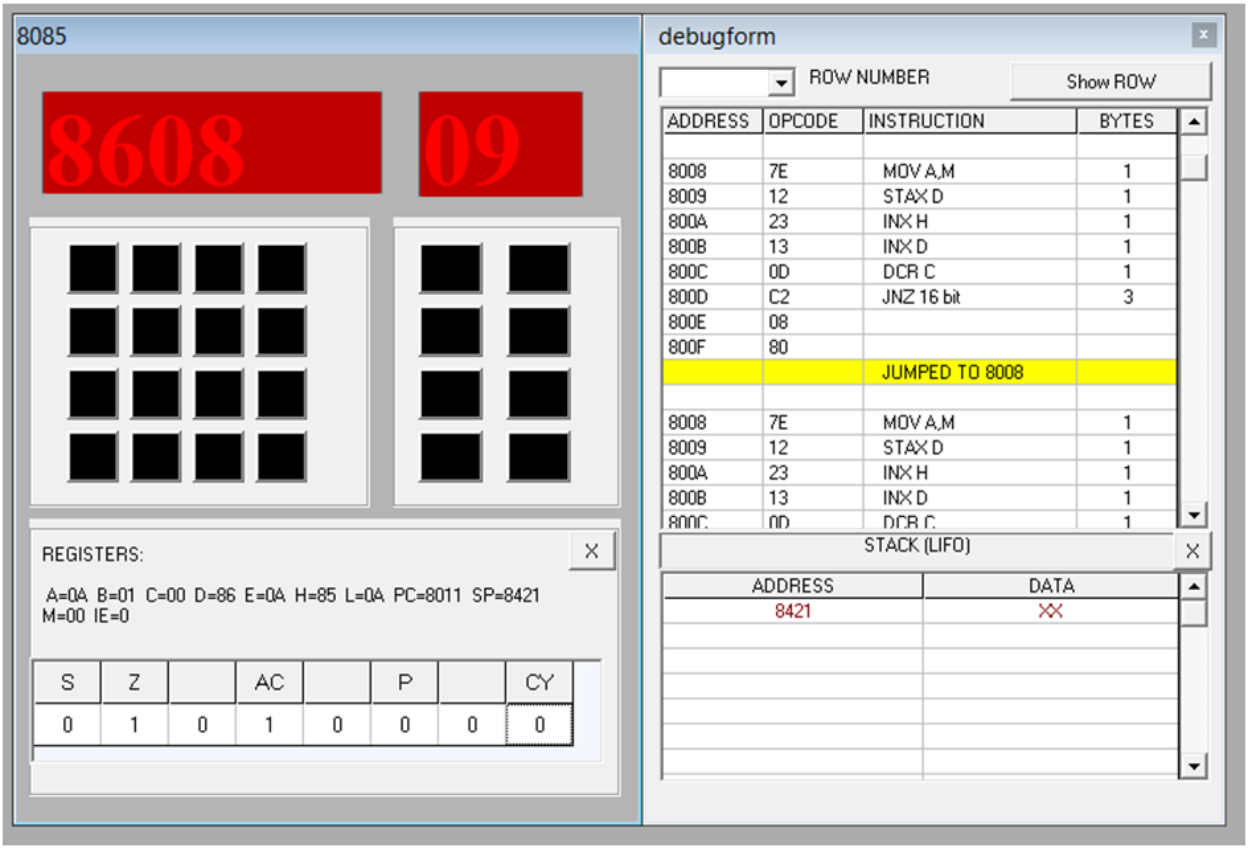

8085
8603
04
REGISTERS:
A=0A B=01 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 1 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
JUMPED TO 8008
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
STACK (LIFO)
ADDRESS DATA
8421 XX

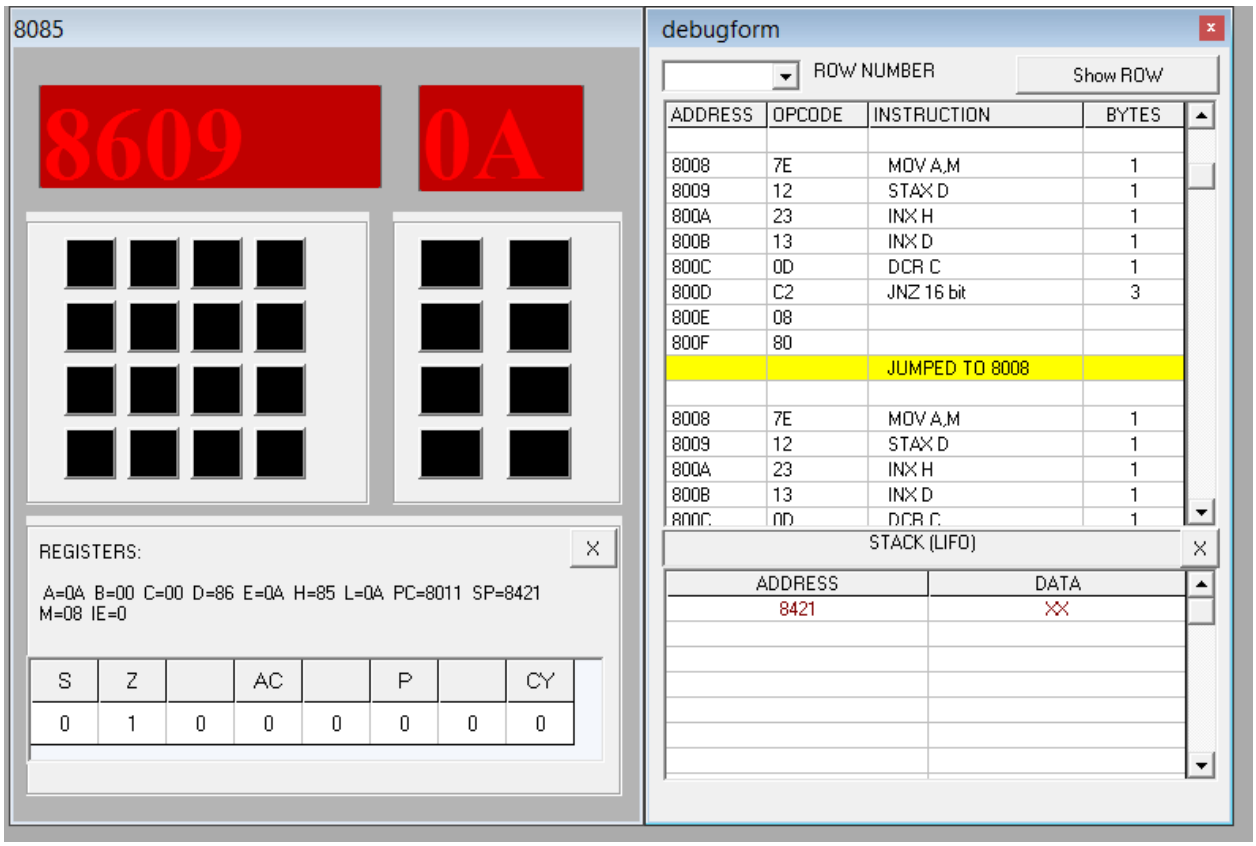

8085
8604
05
REGISTERS:
A=0A B=01 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=00 IE=0
S Z AC P CY
0 1 0 1 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
800C 0D DCR C 1
800D C2 JNZ 16 bit 3
800E 08
800F 80
JUMPED TO 8008
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
8605
06
REGISTERS:
A=0A B=01 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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
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
JUMPED TO 8008
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
8606
07
REGISTERS:
A=0A B=01 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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
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
JUMPED TO 8008
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
8607
08
REGISTERS:
A=0A B=01 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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
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
JUMPED TO 8008
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
8608
09
REGISTERS:
A=0A B=01 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
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
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
JUMPED TO 8008
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
8609
0A
REGISTERS:
A=0A B=00 C=00 D=86 E=0A H=85 L=0A PC=8011 SP=8421
M=08 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
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
JUMPED TO 8008
8008 7E MOV A,M 1
8009 12 STAX D 1
800A 23 INX H 1
800B 13 INX D 1
STACK (LIFO)
ADDRESS DATA
8421 XX

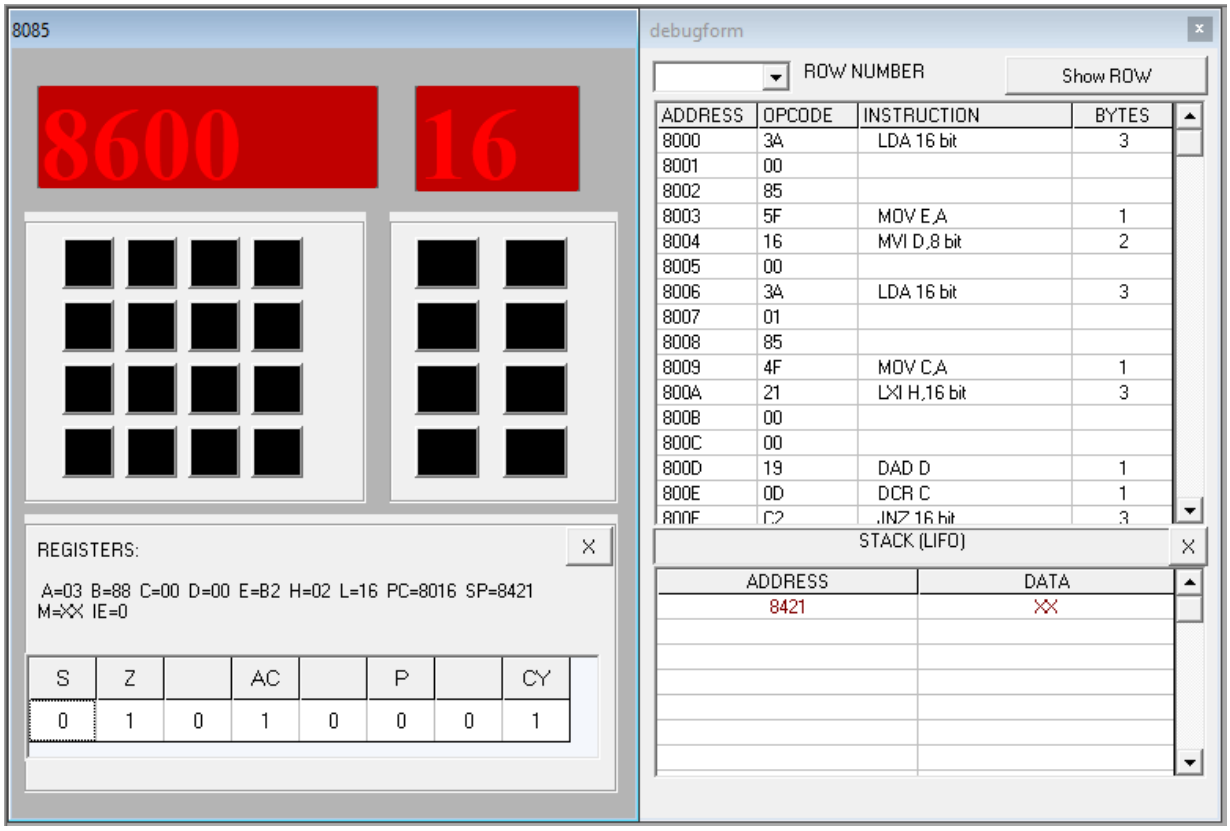
Program 5:

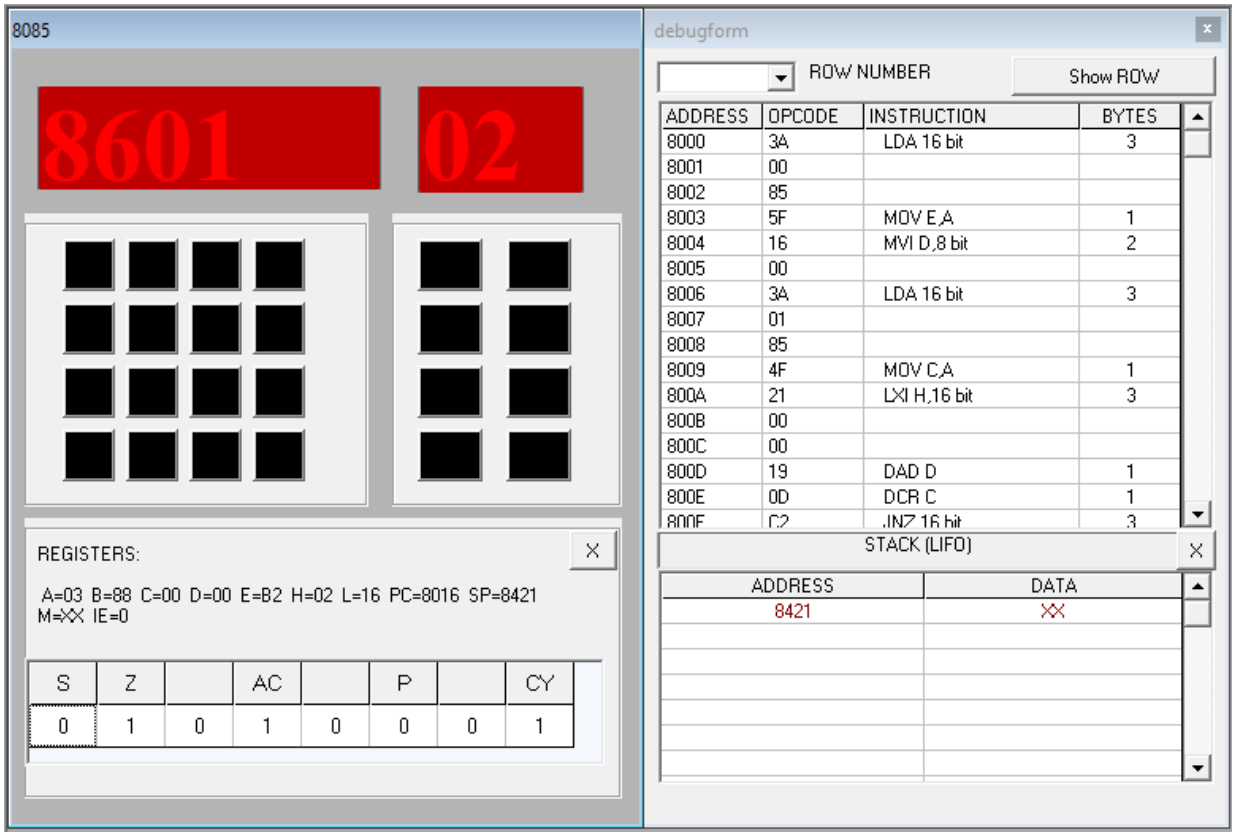
Write a program for multiplying two 8-bit numbers.

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

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

Simulation:


8085
8600
16
REGISTERS:
A=03 B=88 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 1 0 0 0 1
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

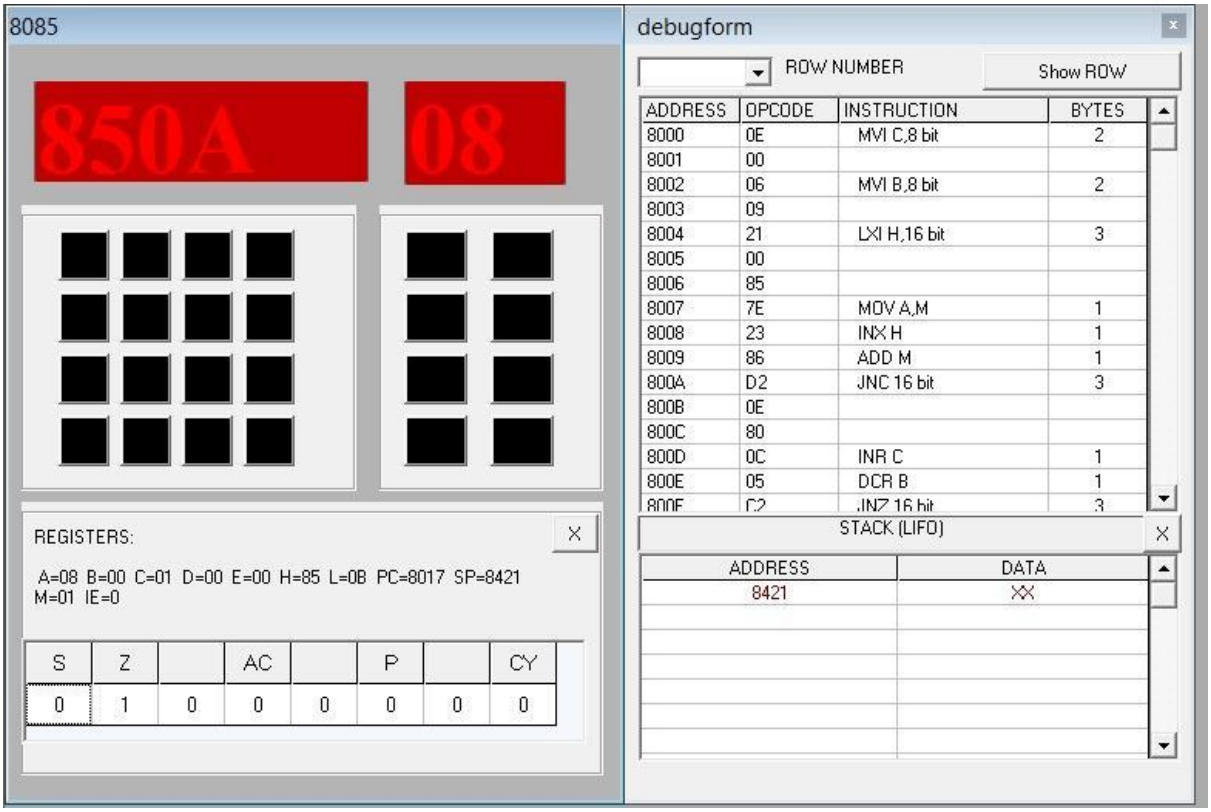

8085
8601
02
REGISTERS:
A=03 B=88 C=00 D=00 E=B2 H=02 L=16 PC=8016 SP=8421
M=XX IE=0
S Z AC P CY
0 1 0 1 0 0 0 1
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
8000 3A LDA 16 bit 3
8001 00
8002 85
8003 5F MOV E,A 1
8004 16 MVI D,8 bit 2
8005 00
8006 3A LDA 16 bit 3
8007 01
8008 85
8009 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

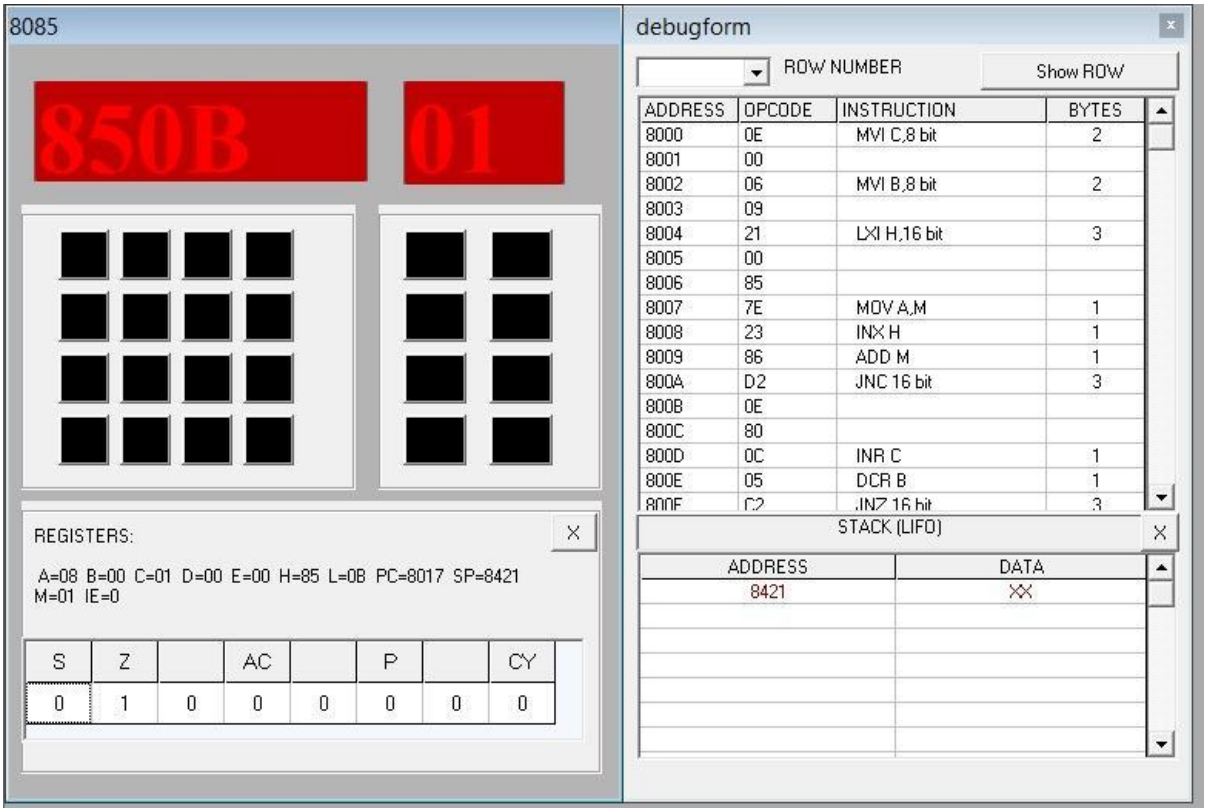
Program 6:
Write a program to add ten 8-bit numbers. Assume the numbers are stored in 8500-8509. Store the result in 850A and 850B memory addresses.

Code	Memory Location	Opcode
MVI C,00	8000, 8001	0E, 00
MVI B,09	8002, 8003	06, 09
LXI H, 8501H	8004, 8005, 8006	21, 00, 85
MOV A, M	8007	4E
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

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
REGISTERS:
A=08 B=00 C=01 D=00 E=00 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


8085
850B
01
REGISTERS:
A=08 B=00 C=01 D=00 E=00 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

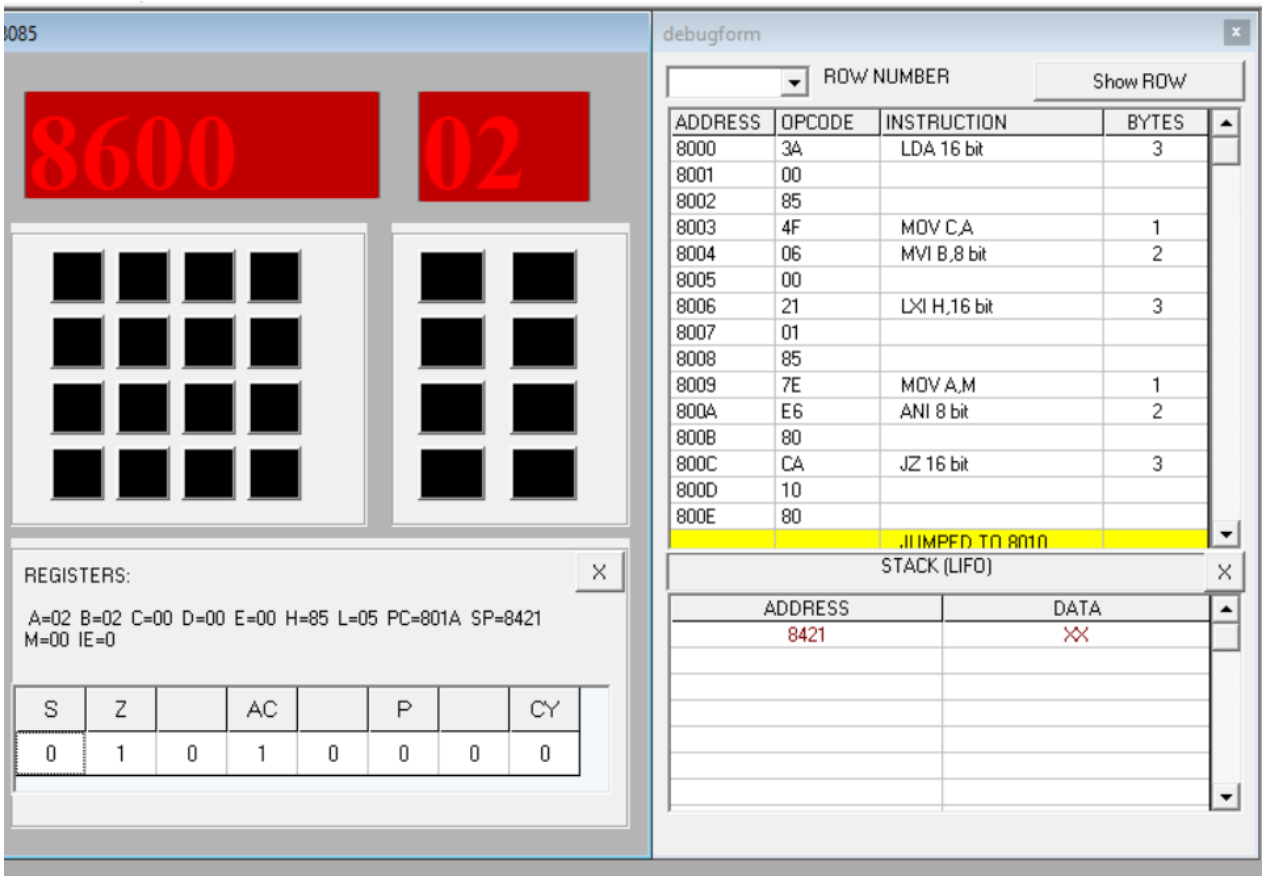
Program 7:

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

Input - [8500] – 04, [8501] – 56, [8502] – A9, [8503] – 73, [8504] – 82
Result - 02
Output - [8600] – 02

Simulation:


8085
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
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)
ADDRESS DATA
8421 XX

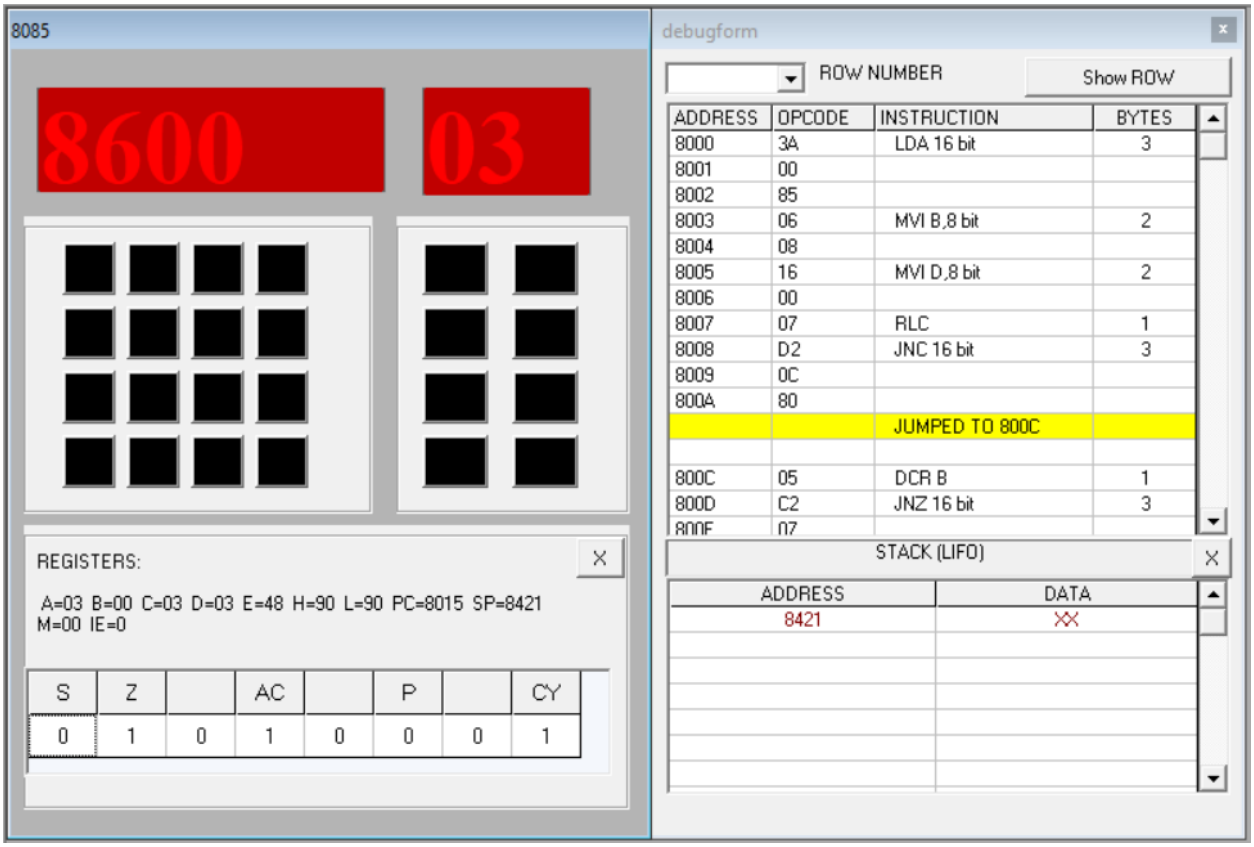
Program 8:
Write a program to count the number of one's in a number.

Code:

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	8014	EF

Simulation:

Input – [8500] - 25
Output – [8600] - 03


8085
8600
03
REGISTERS:
A=03 B=00 C=03 D=03 E=48 H=90 L=90 PC=8015 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 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

Program 9:

Write a program to arrange numbers in Ascending order

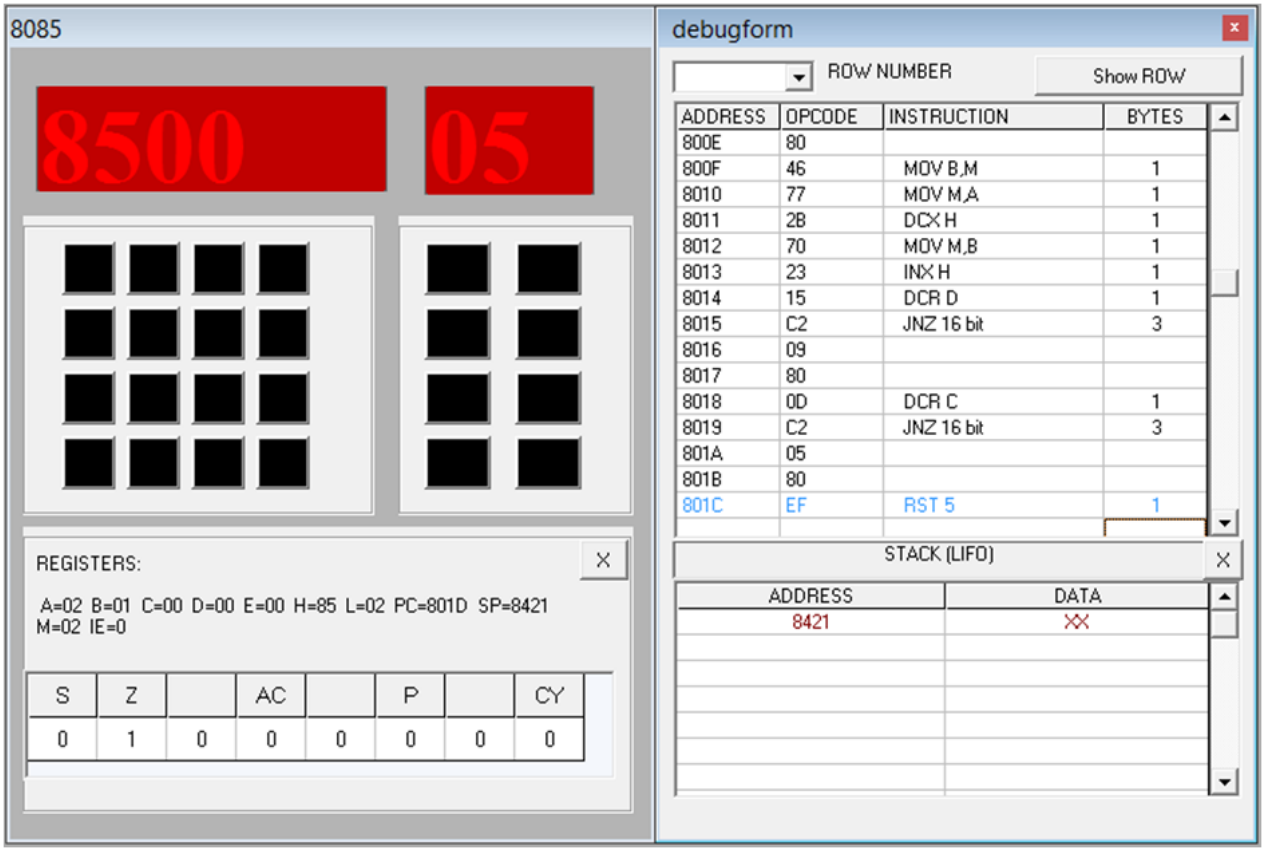
Code	**Memory Location**	**Opcode**
LXI H, 8500H	8000, 8001, 8002	21, 00, 85
MOV C, M	8003	4E
DCR C	8004	0D
Repeat: MOV D, C	8005	51
LXI H, 8501H	8006,8007,8008	21,01,85
Loop: MOV A,M	8009	7E
INX H	800A	23
CMP M	800B	BE
JC Skip	800C,800D,800E	DA,14,80
MOV B, M	800F	46
MOV M, A	8010	77
DCX H	8011	2B
MOV M, B	8012	70
INX H	8013	23

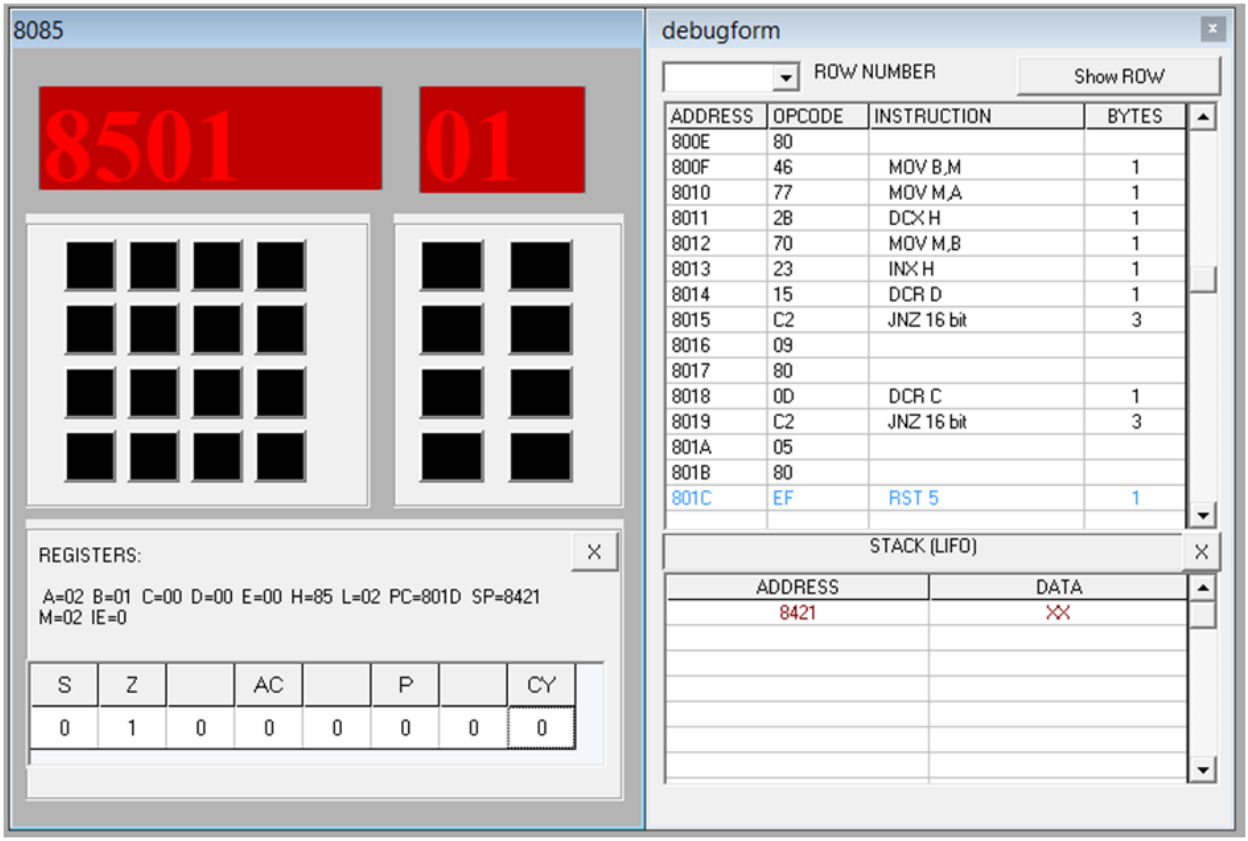
Skip: DCR D	8014	15
JNZ Loop	8015,8016,8017	C2,09,80
DCR C	8018	0D
JNZ Repeat	8019,801A,801B	C2,05,80
RST5	801C	EF

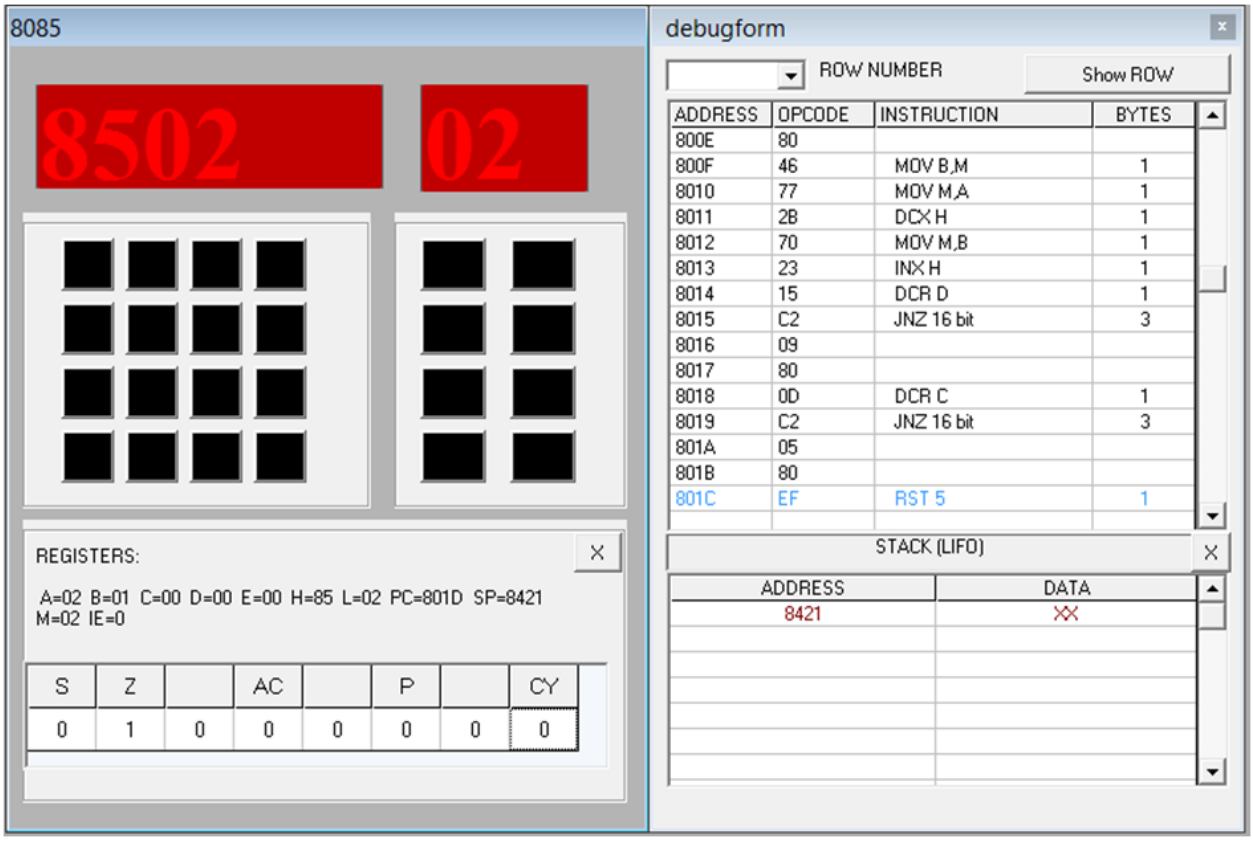
Input - [8500] – 05, [8501] – 05, [8502] – 04, [8503] – 03, [8504] – 02, [8505] – 01

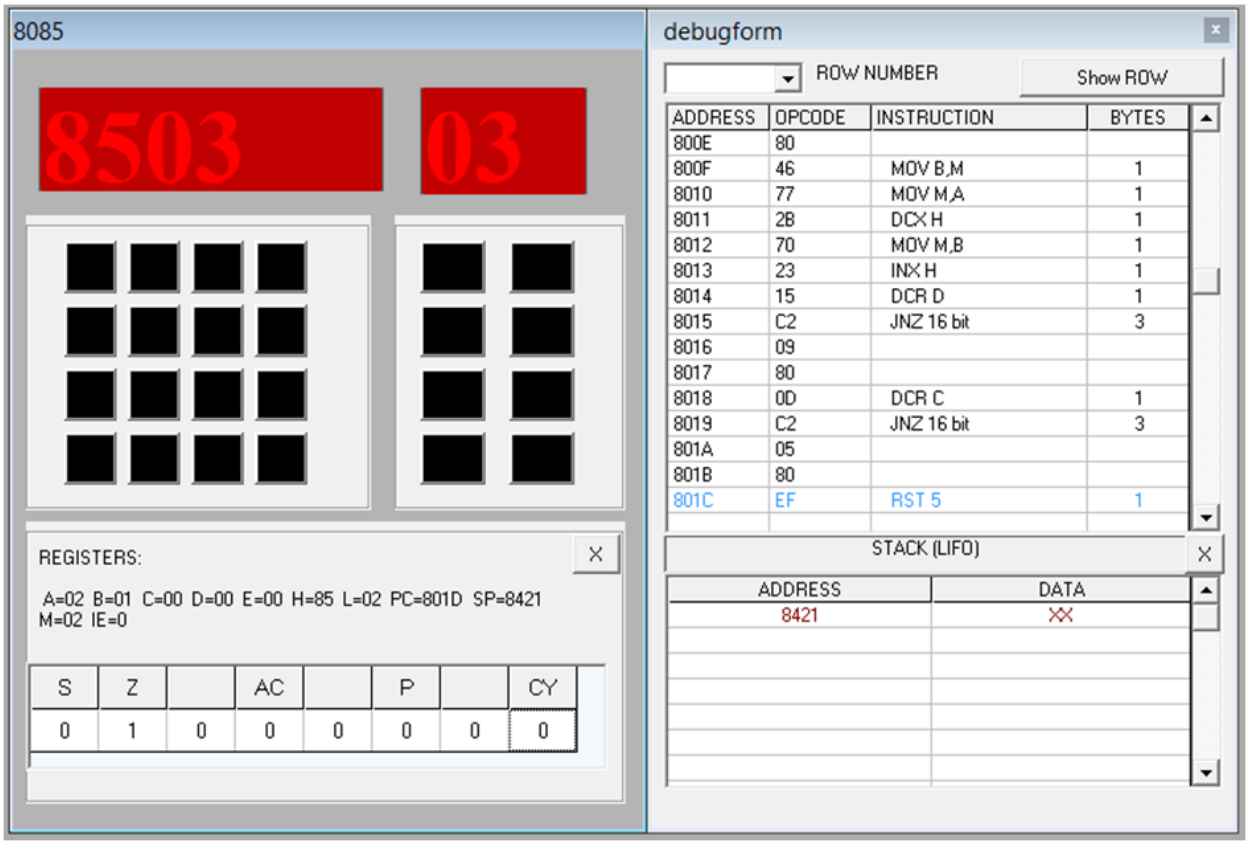
Output - [8500] – 05, [8501] – 01, [8502] – 02, [8503] – 03, [8504] – 04, [8505] – 05

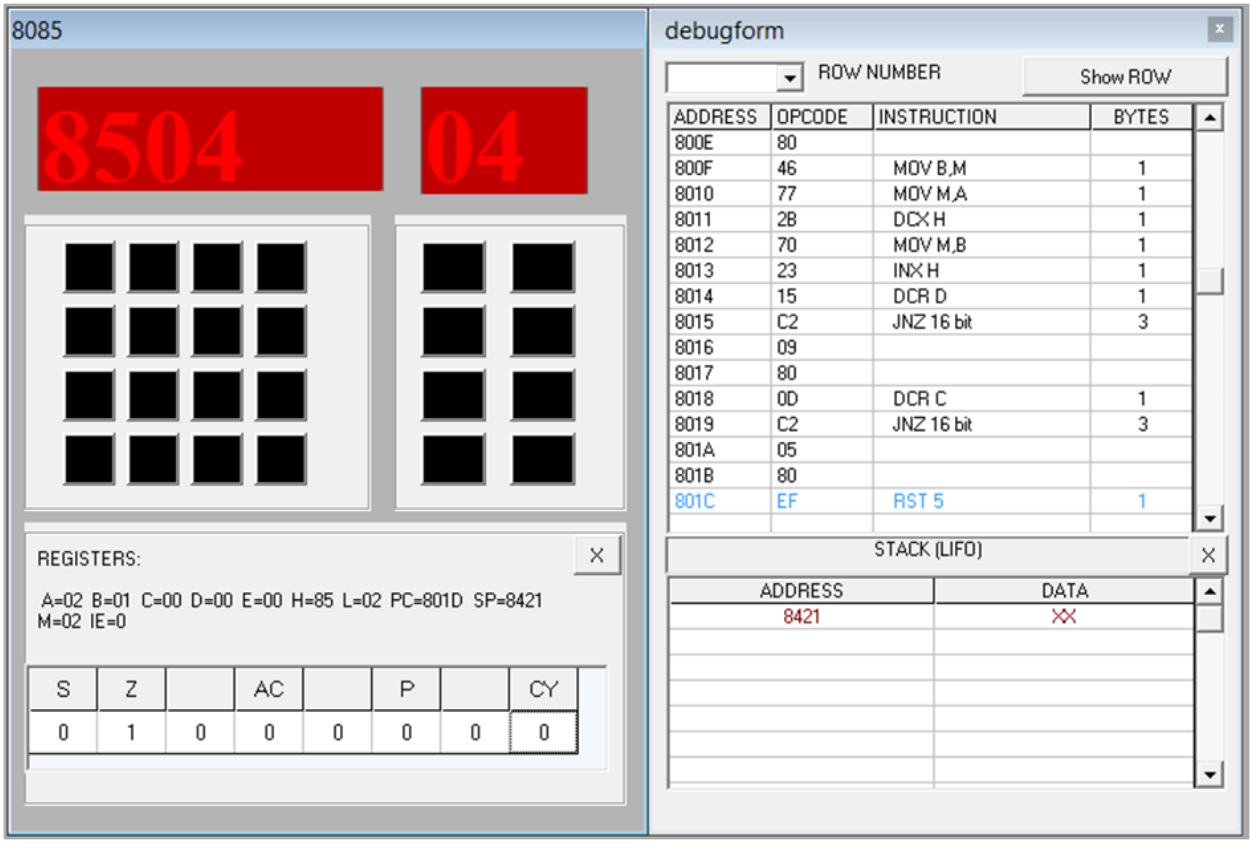
Simulation:

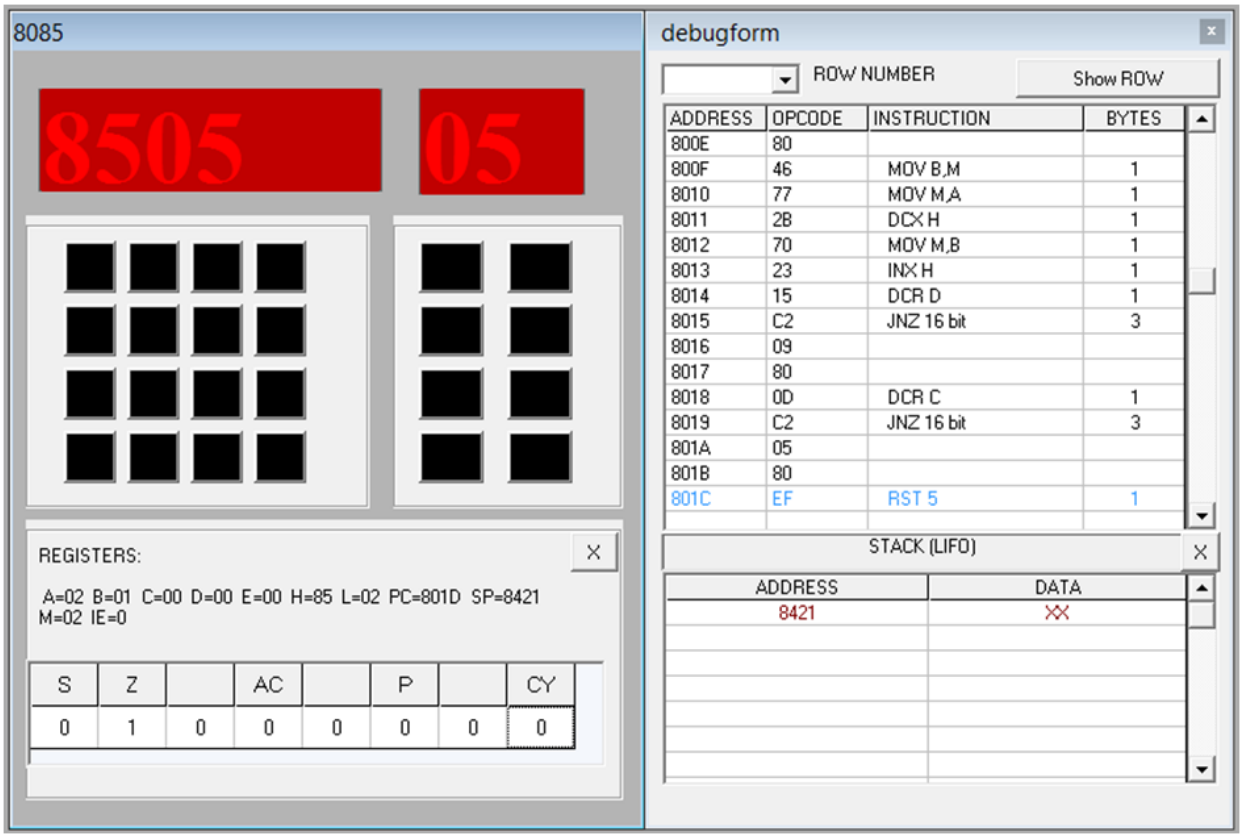



8085
8501
01
REGISTERS:
A=02 B=01 C=00 D=00 E=00 H=85 L=02 PC=801D SP=8421
M=02 IE=0
S Z AC P CY
0 1 0 0 0 0 0 0
debugform
ROW NUMBER
Show ROW
ADDRESS OPCODE INSTRUCTION BYTES
800E 80
800F 46 MOV B,M 1
8010 77 MOV M,A 1
8011 2B DCX H 1
8012 70 MOV M,B 1
8013 23 INX H 1
8014 15 DCR D 1
8015 C2 JNZ 16 bit 3
8016 09
8017 80
8018 0D DCR C 1
8019 C2 JNZ 16 bit 3
801A 05
801B 80
801C EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
8502
02
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
800E 80
800F 46 MOV B,M 1
8010 77 MOV M,A 1
8011 2B DCX H 1
8012 70 MOV M,B 1
8013 23 INX H 1
8014 15 DCR D 1
8015 C2 JNZ 16 bit 3
8016 09
8017 80
8018 0D DCR C 1
8019 C2 JNZ 16 bit 3
801A 05
801B 80
801C EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
8503
03
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
800E 80
800F 46 MOV B,M 1
8010 77 MOV M,A 1
8011 2B DCX H 1
8012 70 MOV M,B 1
8013 23 INX H 1
8014 15 DCR D 1
8015 C2 JNZ 16 bit 3
8016 09
8017 80
8018 0D DCR C 1
8019 C2 JNZ 16 bit 3
801A 05
801B 80
801C EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
8504
04
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
800E 80
800F 46 MOV B,M 1
8010 77 MOV M,A 1
8011 2B DCX H 1
8012 70 MOV M,B 1
8013 23 INX H 1
8014 15 DCR D 1
8015 C2 JNZ 16 bit 3
8016 09
8017 80
8018 0D DCR C 1
8019 C2 JNZ 16 bit 3
801A 05
801B 80
801C EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX


8085
8505
05
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
800E 80
800F 46 MOV B,M 1
8010 77 MOV M,A 1
8011 2B DCX H 1
8012 70 MOV M,B 1
8013 23 INX H 1
8014 15 DCR D 1
8015 C2 JNZ 16 bit 3
8016 09
8017 80
8018 0D DCR C 1
8019 C2 JNZ 16 bit 3
801A 05
801B 80
801C EF RST 5 1
STACK (LIFO)
ADDRESS DATA
8421 XX

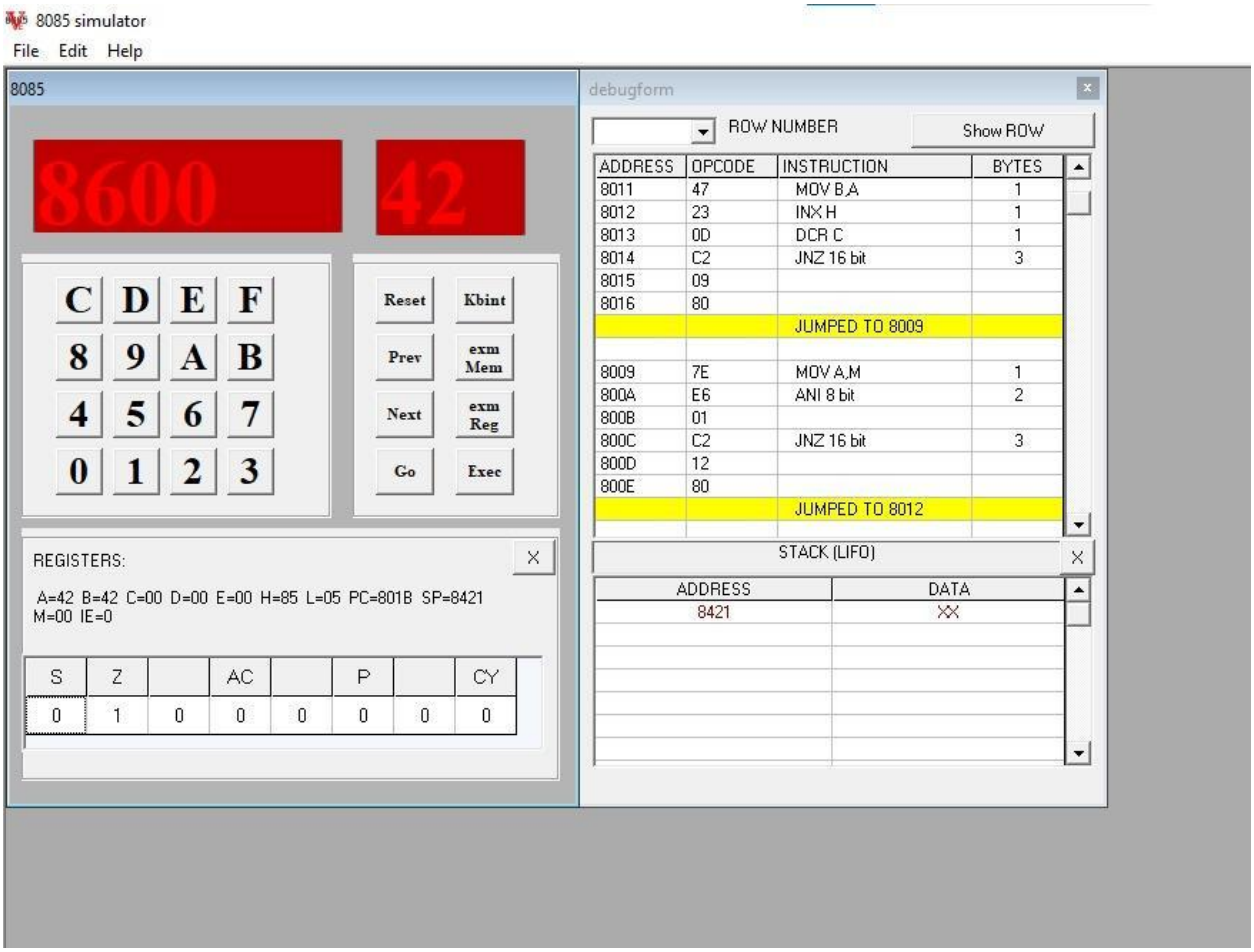
Program 10:
Calculate the sum of a series of even numbers.

Code:

Code	Memory Location	Opcode
LDA 8500	8000,8001,8002	3A,00,85
MOV C,A	8003	4F
MVI B,00	8004,8005	06,00
LXI H,8501	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 8510	8017,8018,8019	32,10,85
RST 5	8010	EF

Simulation:

Input – [8500] - 04, [8501] - 20, [8502] - 15, [8503] - 13, [8504] - 22
Output – [8600] - 42


8085 simulator
File Edit Help
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
8011 47 MOV B,A 1
8012 23 INX H 1
8013 0D DCR C 1
8014 C2 JNZ 16 bit 3
8015 09
8016 80
JUMPED TO 8009
8009 7E MOV A,M 1
800A E6 ANI 8 bit 2
800B 01
800C C2 JNZ 16 bit 3
800D 12
800E 80
JUMPED TO 8012
STACK (LIFO)
ADDRESS DATA
8421 XX

Program 11:

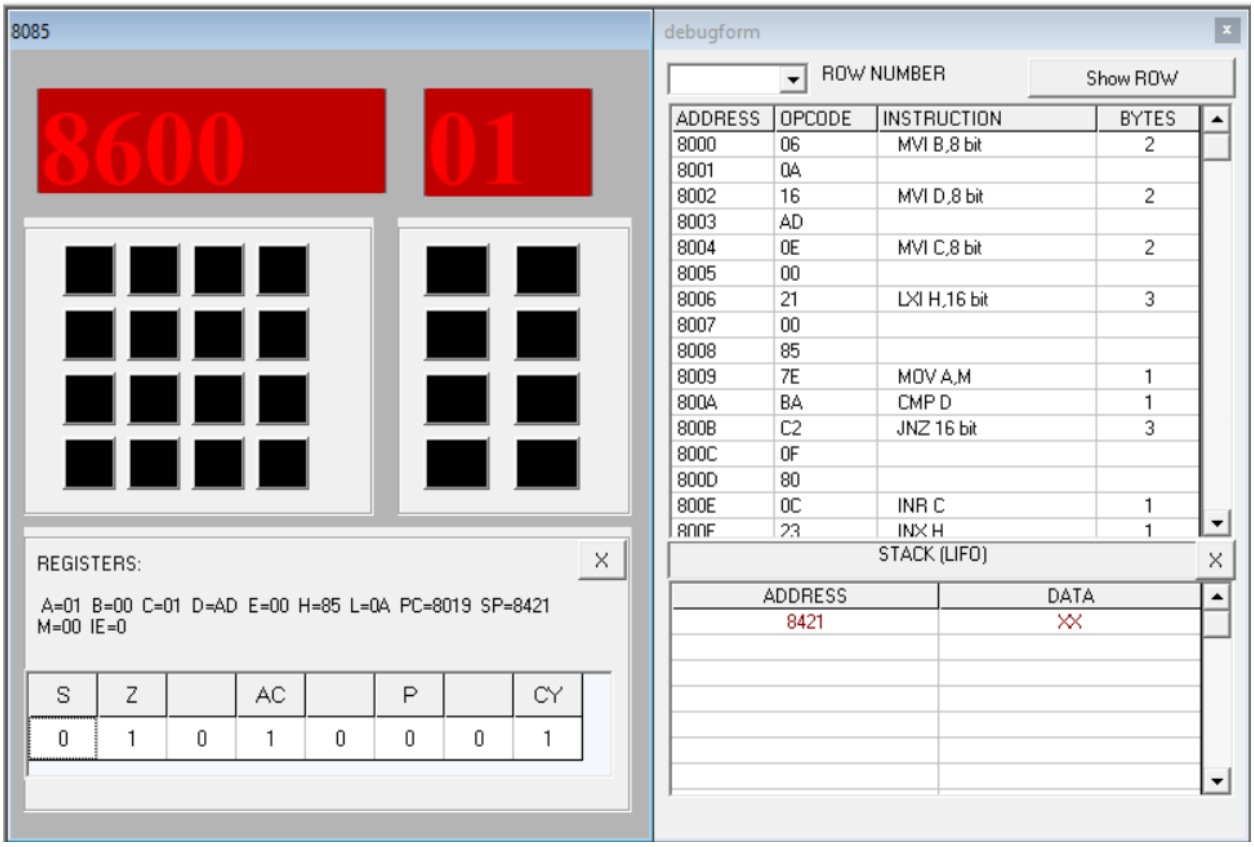
Write an assembly language program to verify how many bytes are present in a given set, which resembles 10101101 in 8085.

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

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

Output - [8600] – 01

Simulation:


8085
8600
01
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

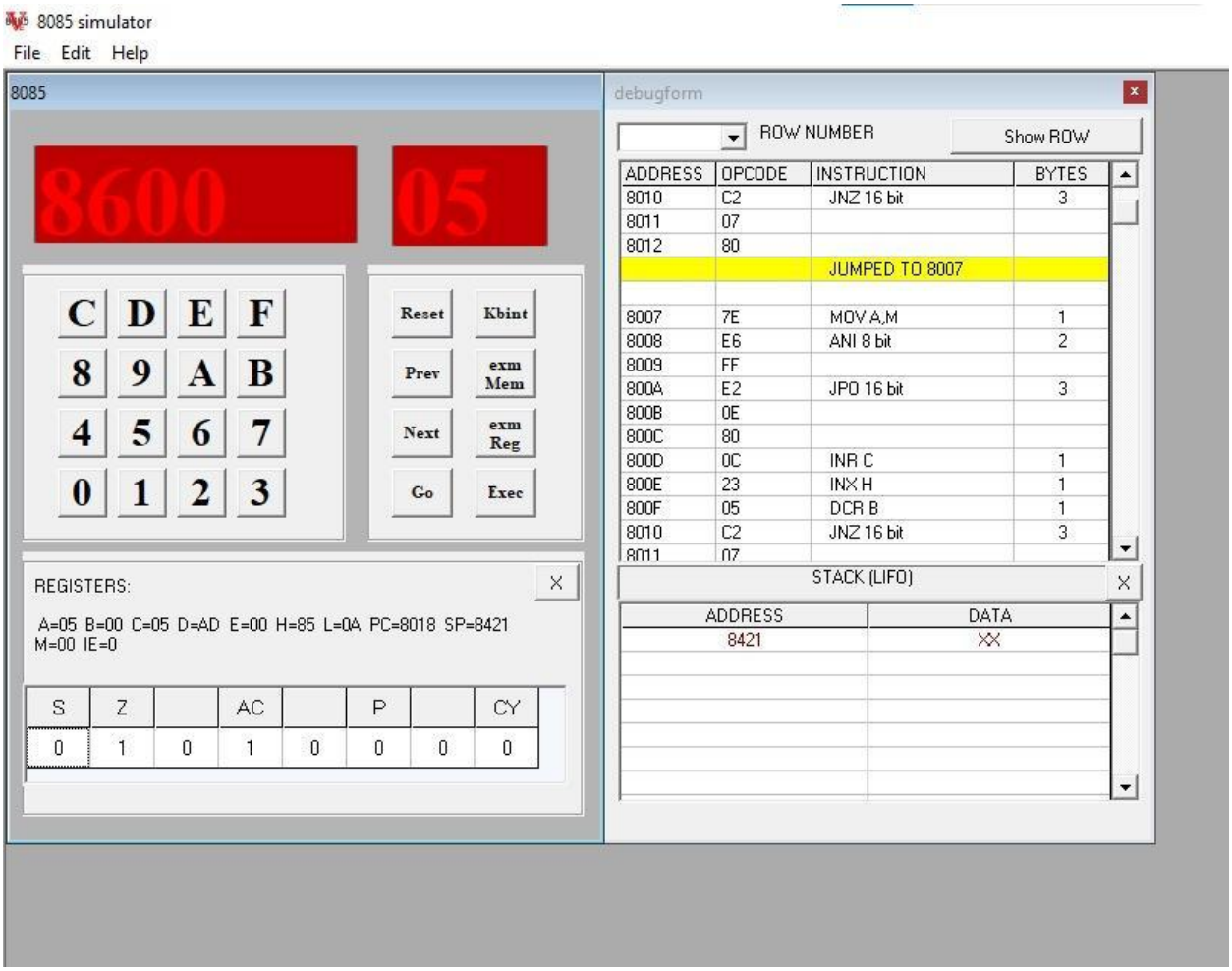
Program 12:
Write an assembly language program to find the numbers of even parity in ten consecutive memory locations in 8085.

Code	Memory Location	Opcode
MVI B, 0A	8000,8001	06,0A
MVI C,00	8002,8003	0E,00
LXI H,8500	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
INC 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 850A	8014,8015,8016	32,0A,85
RST 5	8017	EF

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

Output – [8600] - 05

Simulation:


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=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
8010 C2 JNZ 16 bit 3
8011 07
8012 80
JUMPED TO 8007
8007 7E MOV A,M 1
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
STACK (LIFO)
ADDRESS DATA
8421 XX

Program 13:

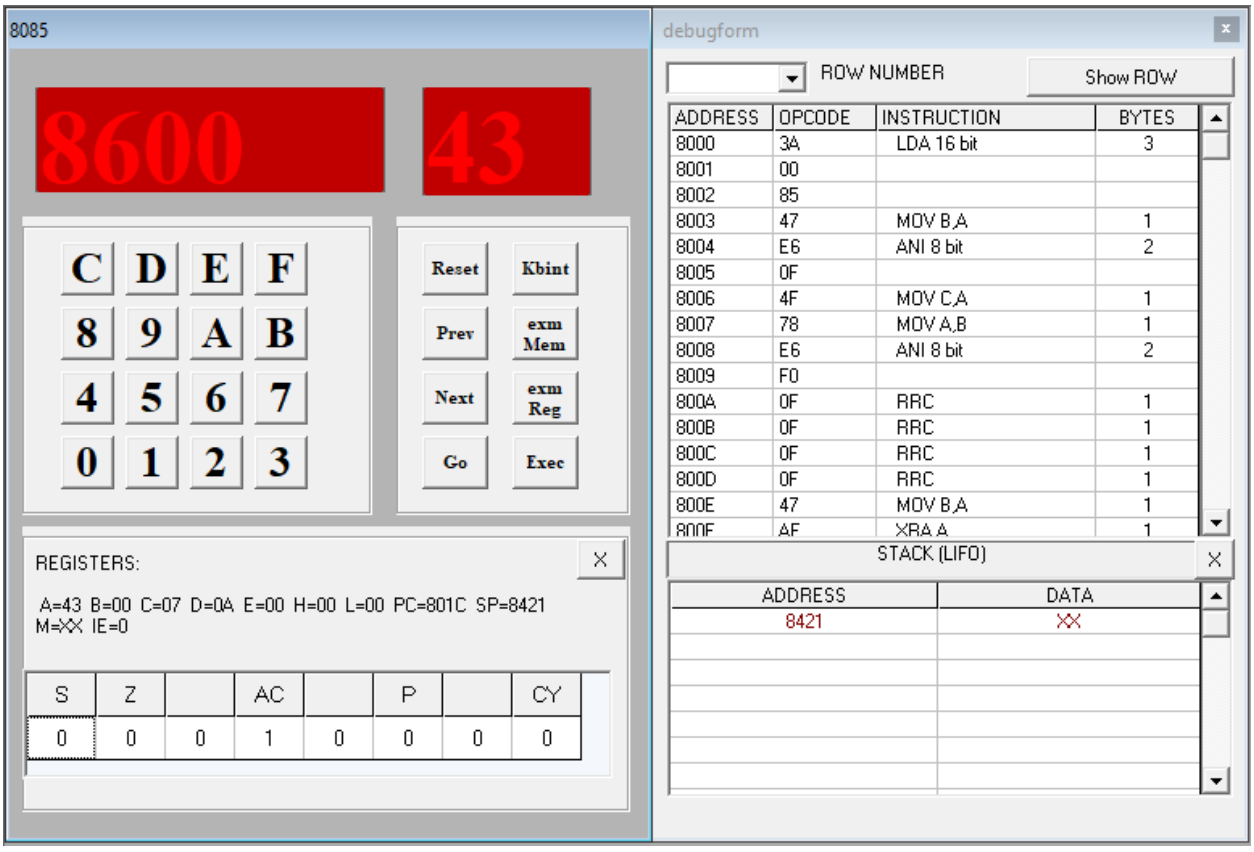
Write an assembly language program to convert a BCD number into its equivalent binary in 8085.

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	16,08
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

Input - [8500] – 67
Output - [8600] – 43

Simulation:



Program 14:

Write an assembly language program 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

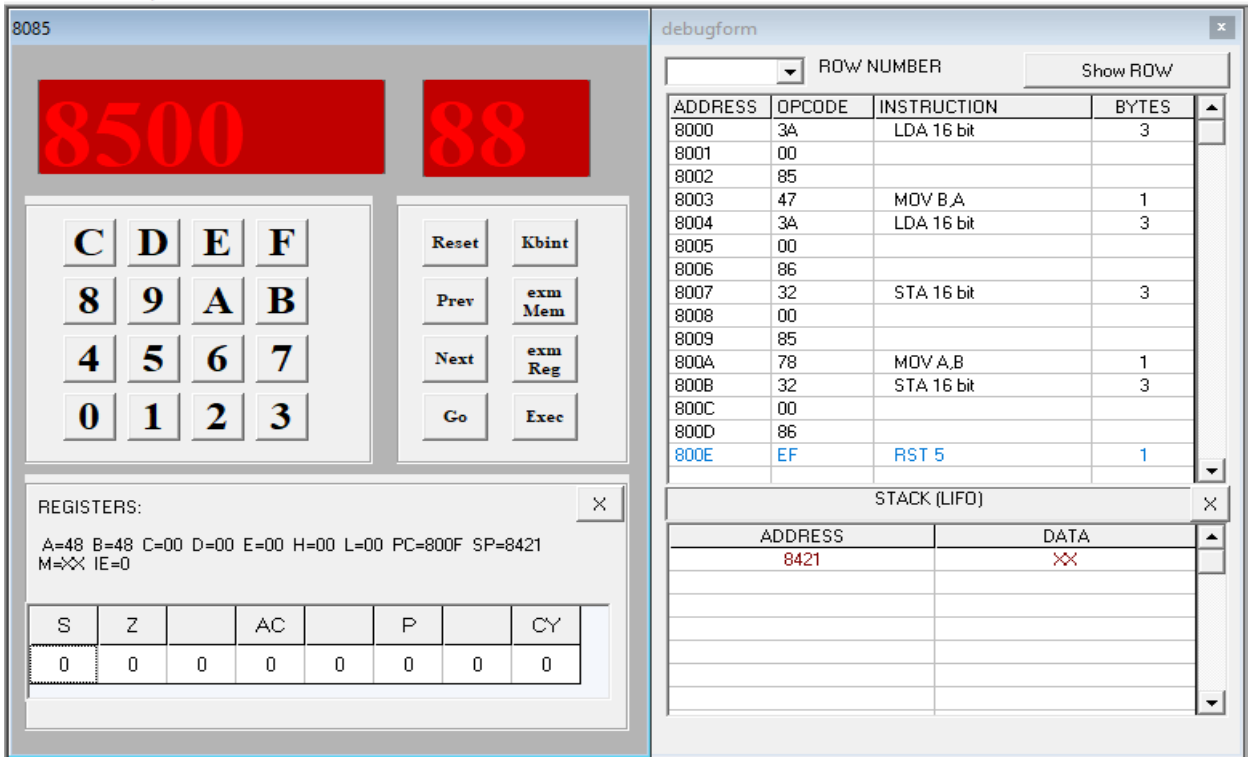
Input - [8500] – 48, [8600] - 88

Output - [8500] - 88, [8600] - 48

Simulation:

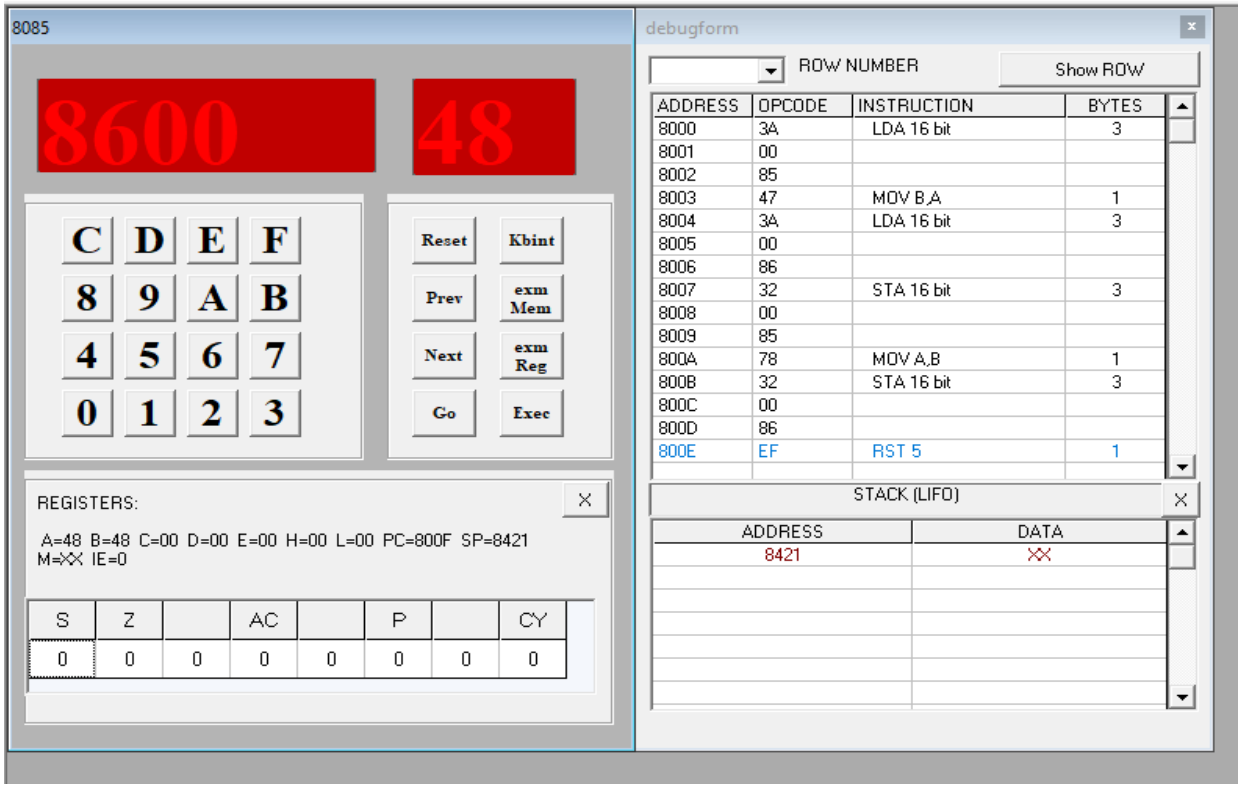
8085 simulator

File Edit Help



8085 simulator

File Edit Help



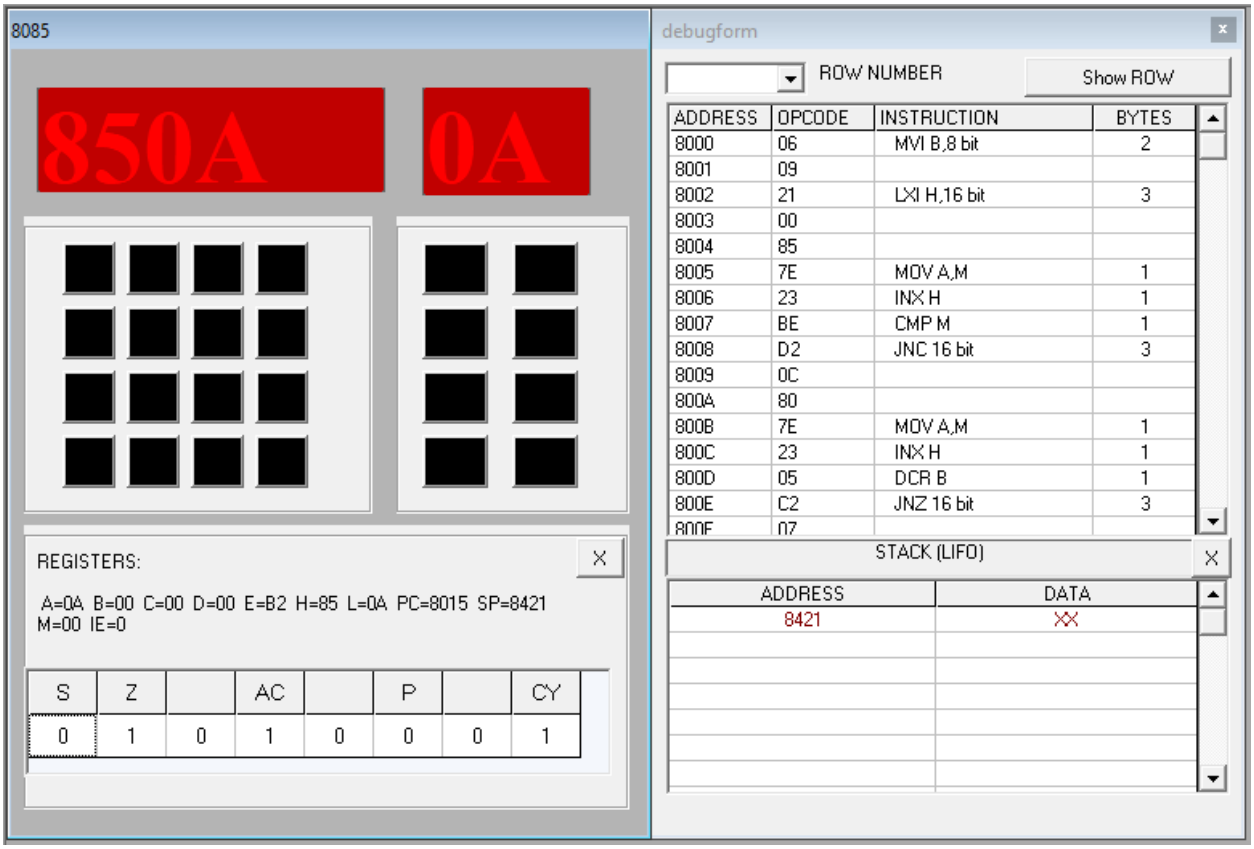
Program 15:

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

Input - [8500] – 01, [8501] – 02… [8509] – 0A
Output - [850A] – 0A

Simulation:


8085
850A
0A
REGISTERS:
X
A=0A B=00 C=00 D=00 E=B2 H=85 L=0A PC=8015 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 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
800F 07
STACK (LIFO)
X
ADDRESS DATA
8421 XX