MICROPROCESSOR-BASED SYSTEMS DESIGN

UCS617

Lab Assignment-1 (8085)

Submitted to:

Dr Manju Khurana

Submitted By:

Dixant Kumar 102103017

Jagdish Agarwal 102103266

Pulkit Arora 102103267

Jeetesh Rajpal 102103268

Janardhan Singh Jadon 102103269

BE Third Year, COE

Group No: 3COE10

Computer Science and Engineering Department

TIET, Patiala

INDEX

Experiment 1

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

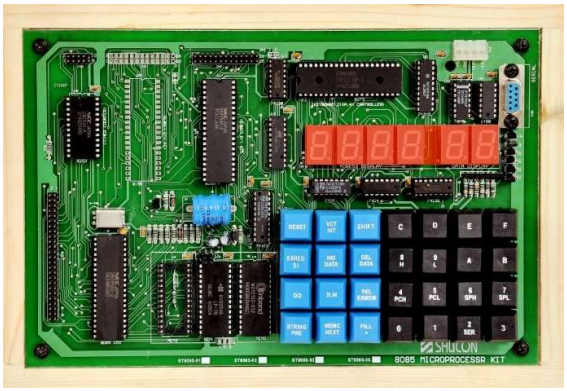


Figure 1.1 Intel 8085 Microprocessor

Features of 8085 Microprocessor

1. 8085 is developed by INTEL.
2. 8-bit microprocessor: can accept 8-bit data simultaneously.
3. Operates on single +5V D.C. supply.
4. Designed using NMOS technology.
5. 6200 transistors on a single chip.
6. It provides an on-chip clock generator, it does not require an external clock generator.
7. Operates on 3MHz clock frequency.
8. 8-bit multiplexed address/data bus, which reduces the number of pins.
9. 16 address lines, hence it can address 2^16 = 64 K bytes of memory
10. It generates 8 bit I/O addresses; hence it can access 2^8 = 256 I/O ports.
11. 5 hardware interrupts i.e. TRAP/RST4.5, RST 7.5, RST 6.5, RST 5.5, and INTR
12. It provides DMA (Direct memory access).
13. 40-pin I.C. package fabricated on a single LSI chip.
14. Clock cycle is 320ns.
15. 80 basic instructions and 246 opcodes.

The 8085 microprocessors, features a 40-pin dual in-line package (DIP) configuration, with pins categorized into power supply, address bus, data bus, and control bus groups. The address bus comprises 16 lines (A0-A15), enabling a maximum memory capacity of 64 KB, while the 8-bit data bus (D0-D7) facilitates data transfer. Employing multiplexing, the microprocessor optimizes pin usage by transmitting lower and higher order address bytes during consecutive clock cycles. Various registers, including the accumulator, general-purpose (B, C, D, E, H, L), and special-purpose (stack pointer, program counter) registers, play integral roles in data manipulation and program control. Interrupt handling is a key feature, supporting TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR interrupts, each with specific vector addresses and prioritized responses. The interrupt enable/disable (IE) flag regulates responses, while the INTR pin manages external interrupt signals.

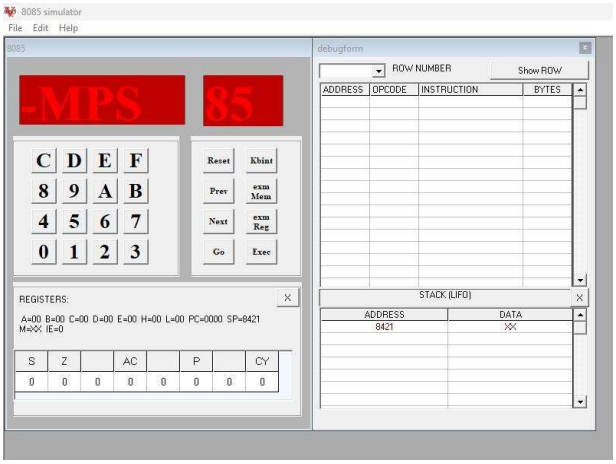


Figure 1.2 Vikas Simulator

Functions

1. **Reset:** Initializes the simulator to its default state.
2. **Prev:** Steps back to the previous instruction or state in program execution.
3. **Next**: Advances to the next instruction or state in program execution.
4. **Go:** Initiates the execution of the program until a breakpoint or completion.
5. **Exec**: Executes the current instruction and proceeds to the next one.
6. **Exm Reg**: Examines the contents of registers in the simulation.
7. **Exm Mem:** Examines the contents of memory locations in the simulation.

Steps to Execute on Vikas Simulator:

1. Press Reset
2. Press Examine Memory
3. Enter starting address.
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

Program No. 2.1

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

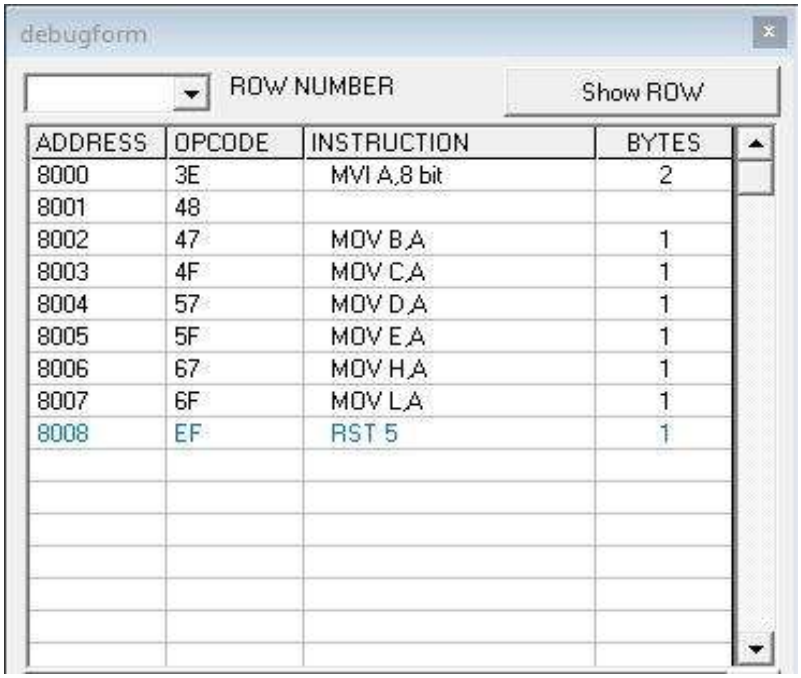


Figure 2.1.1

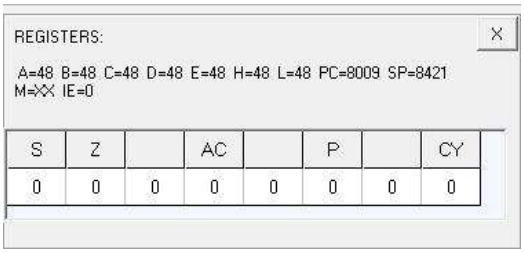


Figure 2.1.2 Values of Registers

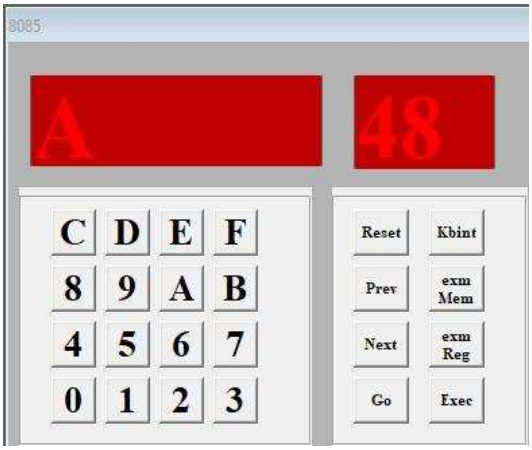


Figure 2.1.3 [A] - 48

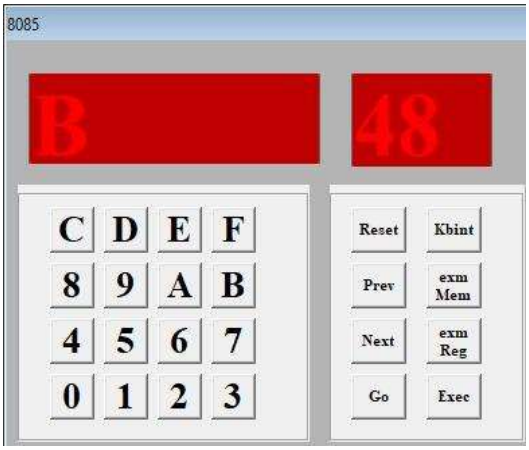


Figure 2.1.4 [B] - 48

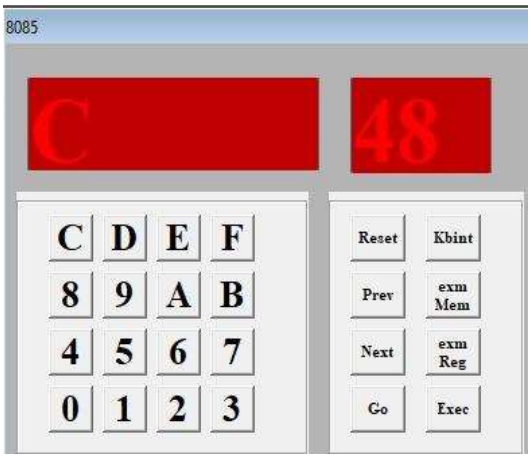


Figure 2.1.5 [C] - 48

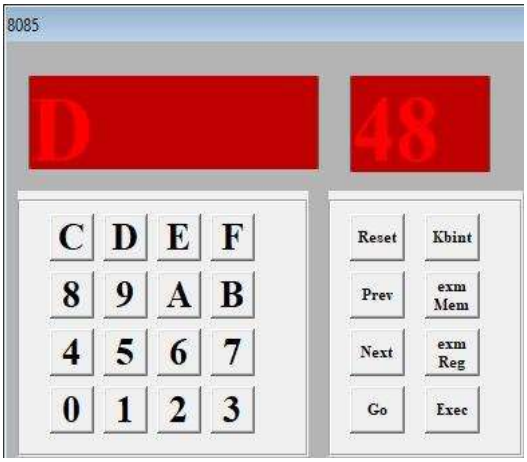


Figure 2.1.6 [D] - 48

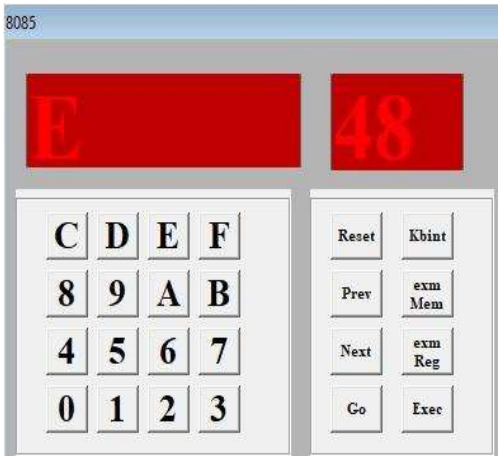


Figure 2.1.7 [E] - 48

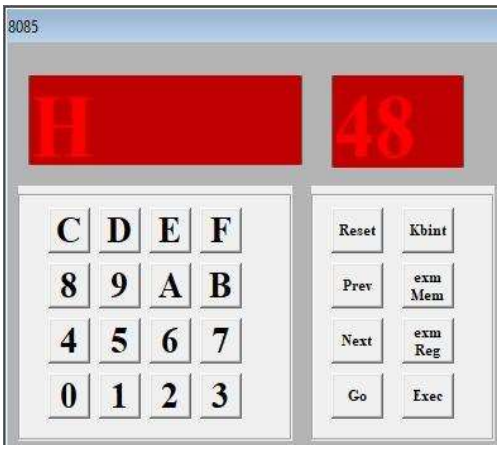


Figure 2.1.8[H] - 48

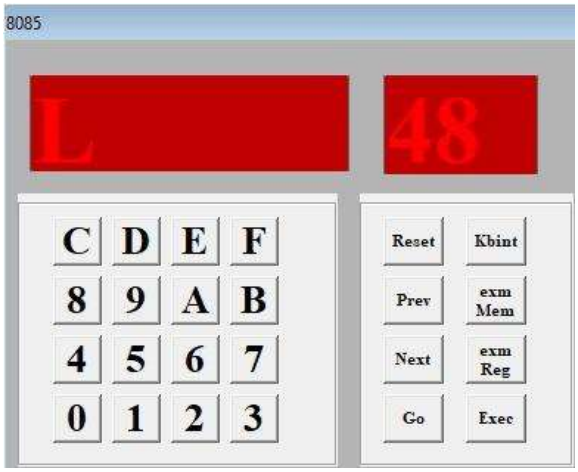


Figure 2.1.9 [L] - 48

Output

Registers: [A] – 48, [B] – 48, [C] – 48, [D] – 48, [E] – 48, [H] – 48, [L] – 48

Program No. 2.2

Aim: 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 program for addition of two 8-bit numbers.

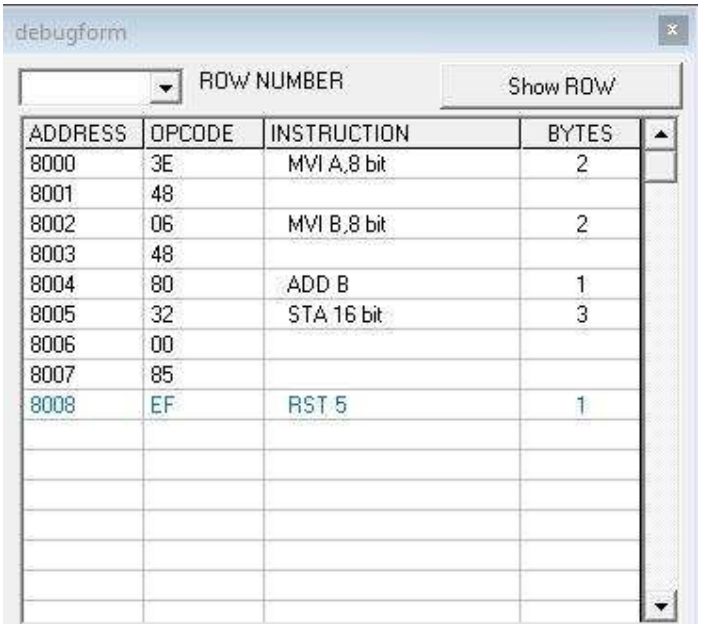


Figure 2.2.1 Addition of two 8-bit numbers.

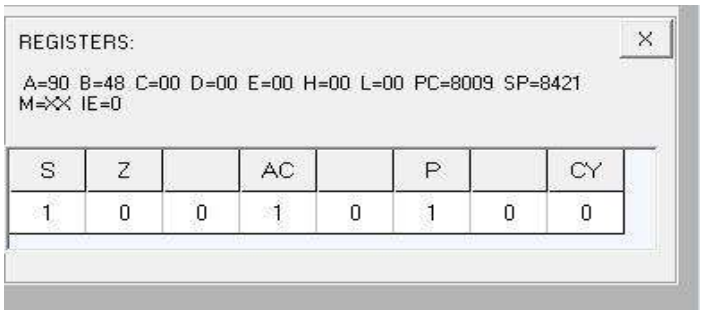


Figure 2.2.2 Values of Registers

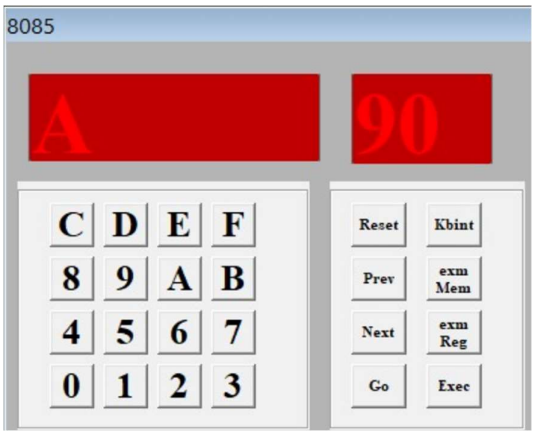


Figure 2.2.3 [A] – 90

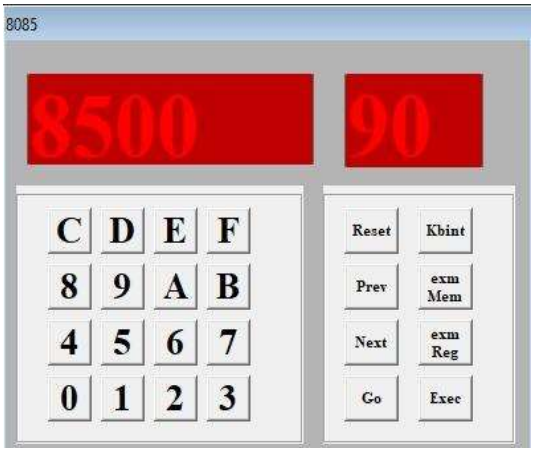


Figure 2.2.4 [8500] - 90

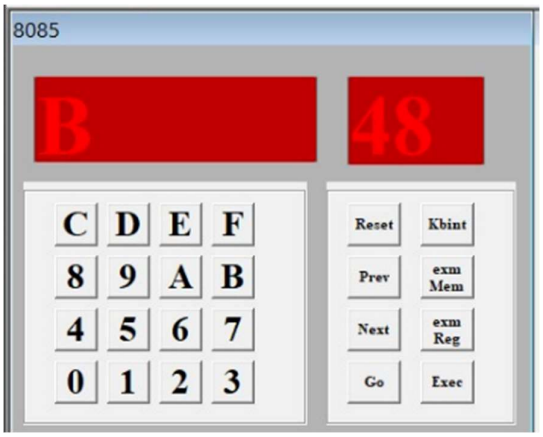


Figure 2.2.5 [B] - 48

Output

[8500] – 90

Program No. 2.3

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

A. 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 2.3.1 Program to add 8-bit numbers using direct addressing mode.

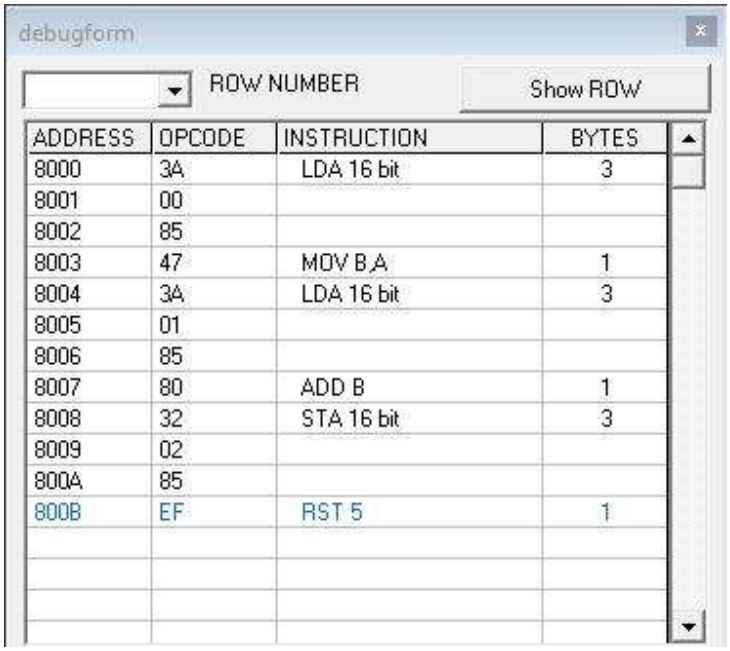


Figure 2.3.1 Add 8-bit numbers using direct addressing mode

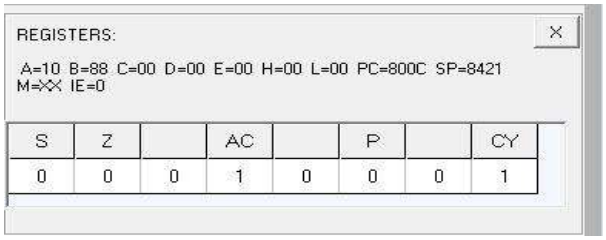


Figure 2.3.2 Values of Registers

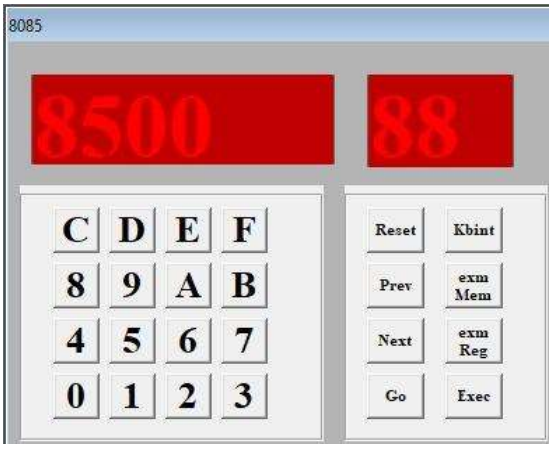


Figure 2.3.3 [8500] - 88

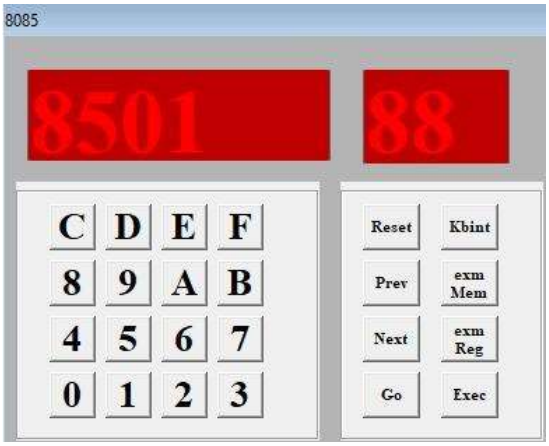


Figure 2.3.4 [8501] - 88

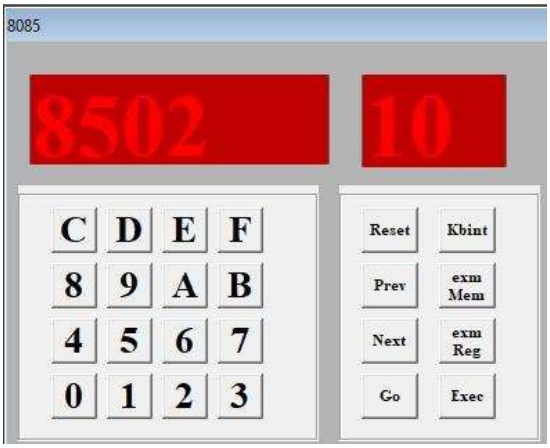


Figure 2.3.5 [8502] - 10

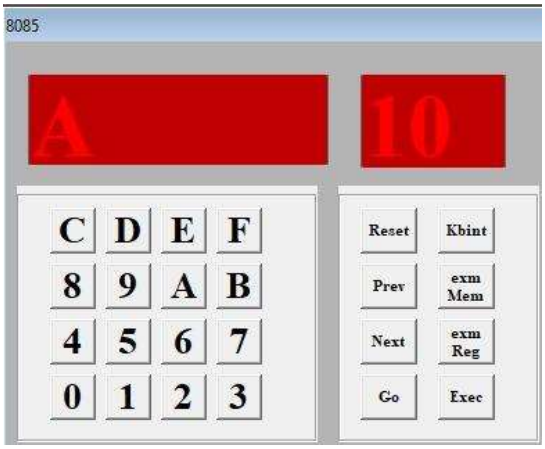


Figure 2.3.6 [A] - 10

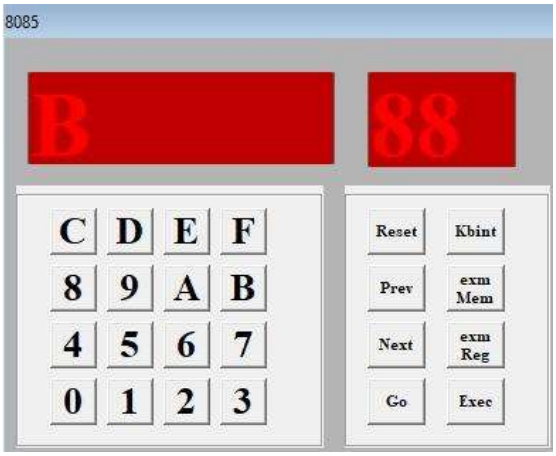


Figure 2.3.7 [B] - 88

Input

[8500] – 88, [8501] – 88

Output

[8502] – 10

B. 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 2.3.2 Program to add 8-bit numbers using indirect addressing mode.

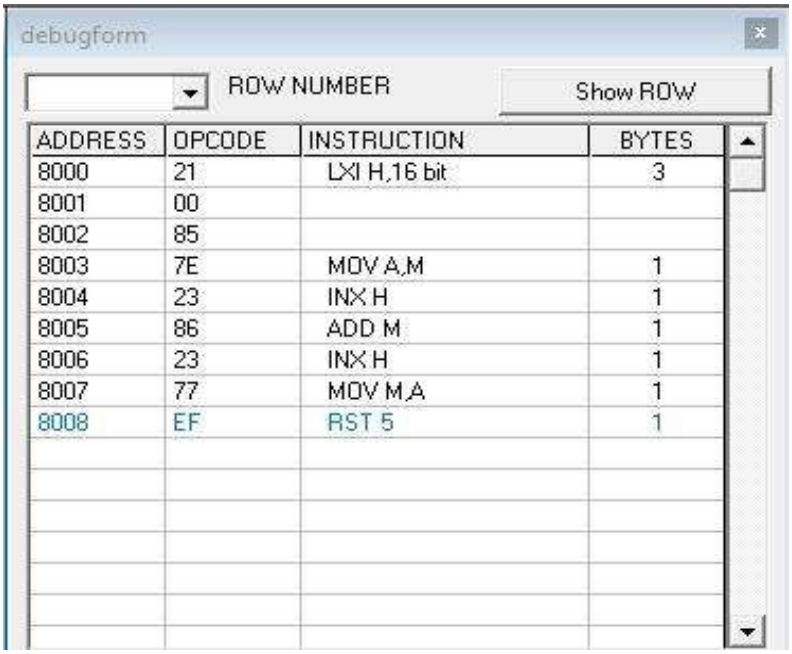


Figure 2.3.8 add 8-bit numbers using indirect addressing mode.

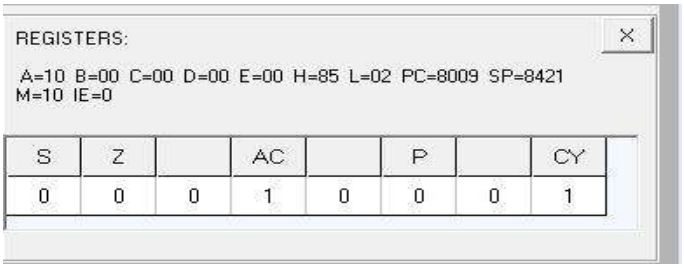


Figure 2.3.9 Values of Registers

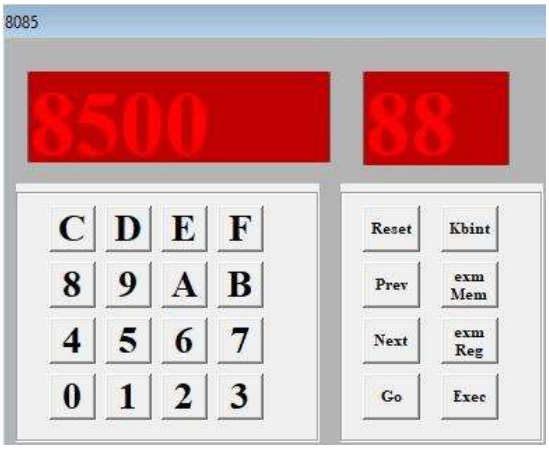


Figure 2.3.10 [8500] - 88

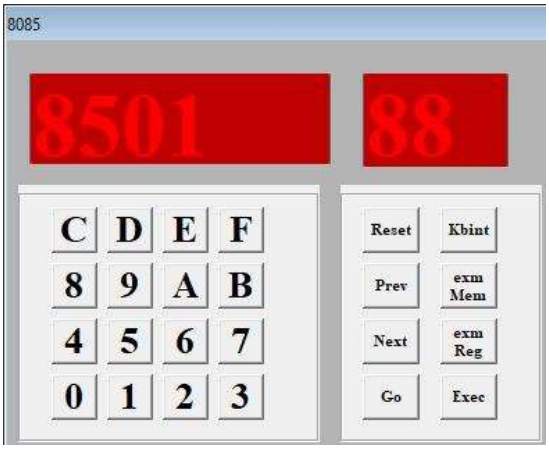


Figure 2.3.11 [8501] - 88

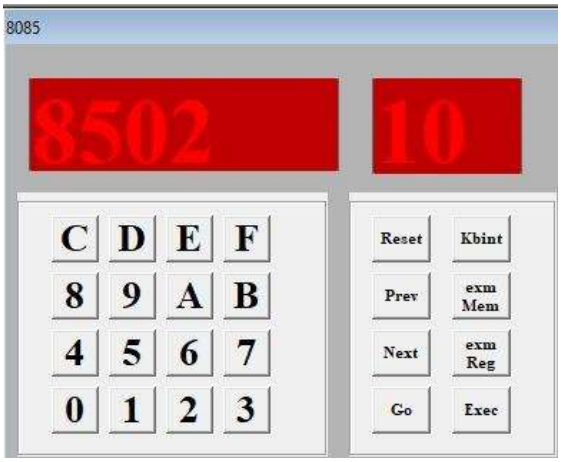


Figure 2.3.12 [8502] - 10

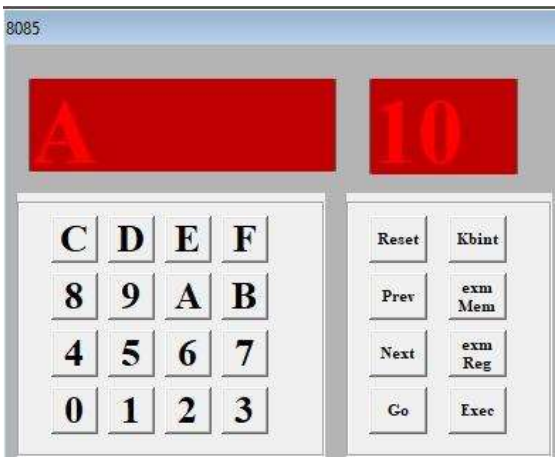


Figure 2.3.13 [A] -10

Input

[8500] – 88, [8501] – 88

Output

Registers: [A] – 10, [H] – 85, [L] - 02

Memory Addresses: [8500] – 88, [8501] – 88, [8502] - 10

Flags: S – 0, Z – 0, AC – 1, P – 0, CY - 1

Program No. 2.4

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

A. 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 2.4.1 program to add 16-bit numbers using direct addressing mode.

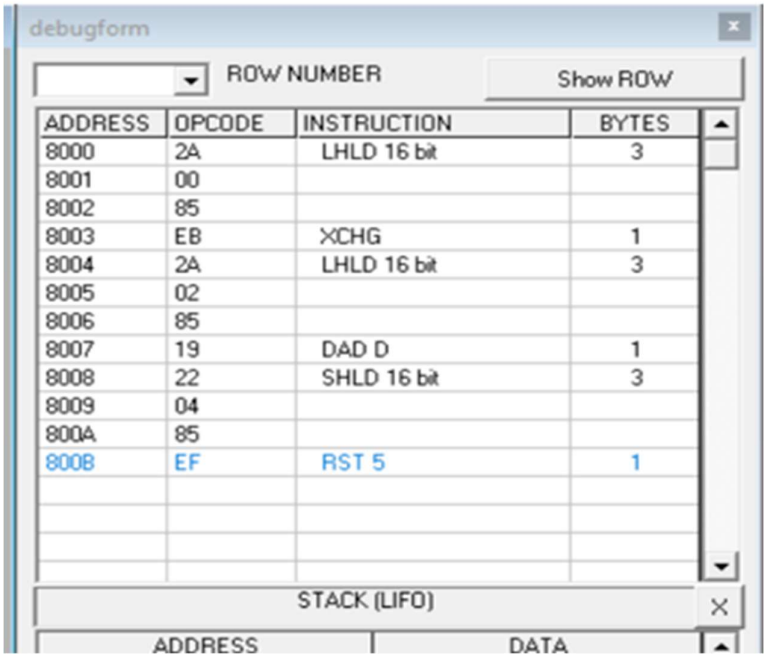


Figure 2.4.1 add 16-bit numbers using direct addressing mode.

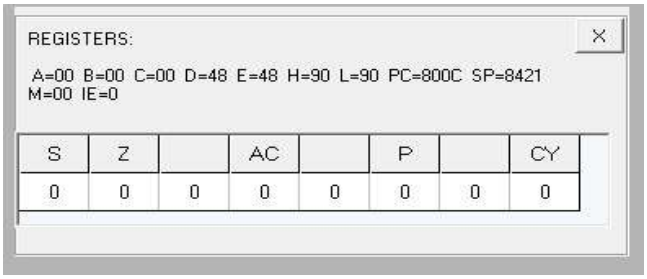


Figure 2.4.2 Values of Registers

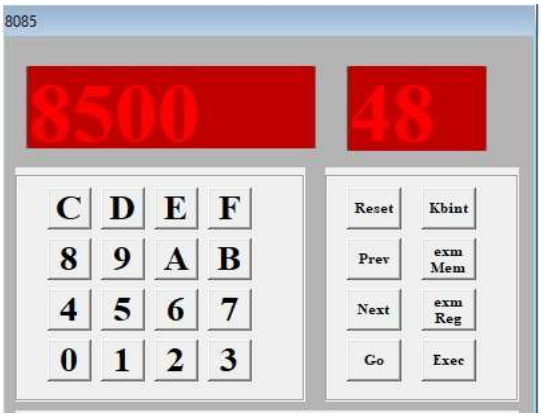


Figure 2.4.3 [8500] - 48

Figure 2.4.4 [8501] - 48

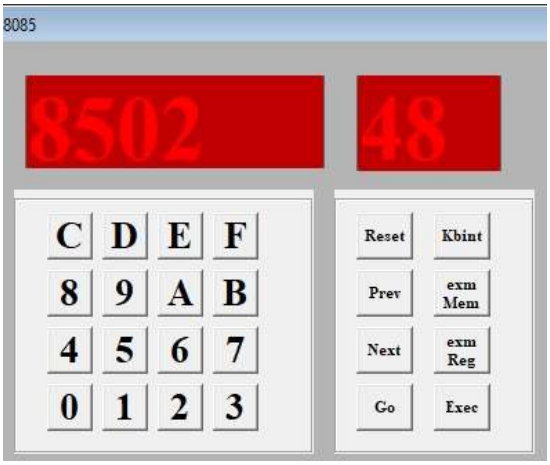


Figure 2.4.5 [8502] - 48

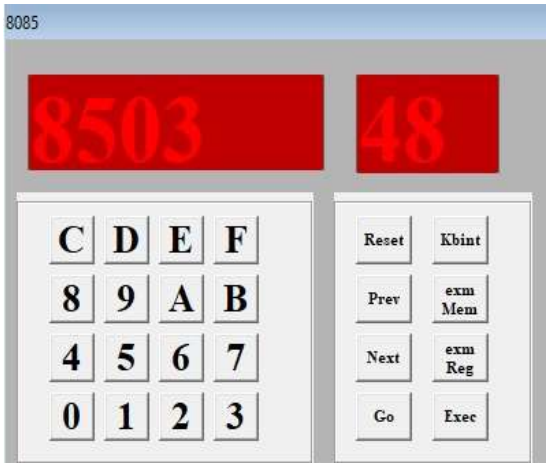


Figure 2.4.6 [8503] - 48

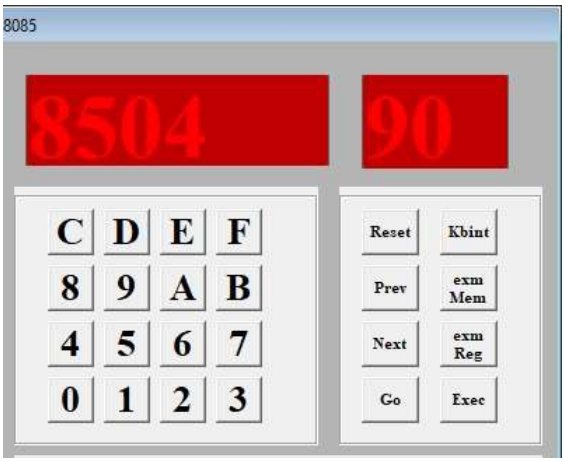


Figure 2.4.7 [8500] - 90

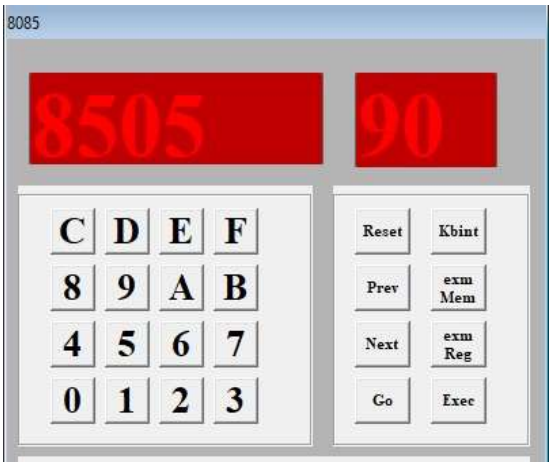


Figure 2.4.8 [8500] - 90

Input

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

Output

Registers: [D] – 48, [E] – 48, [H] – 90, [L] - 90

Memory Address: [8504] - 90, [8505] - 90

Flags: S – 0, Z – 0, AC – 0, P – 0, CY – 0

B. 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 2.4.2 program to add 16-bit numbers using indirect addressing mode.

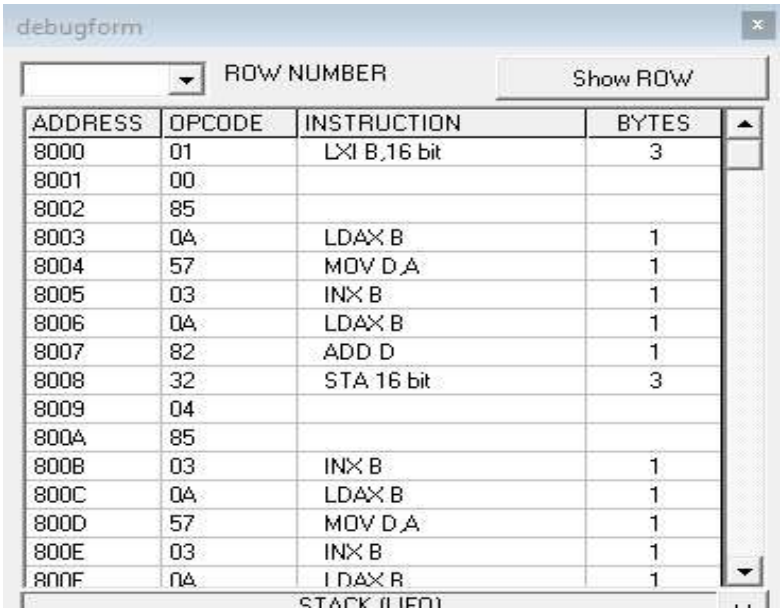


Figure 2.4.8 add 16-bit numbers using indirect addressing mode.

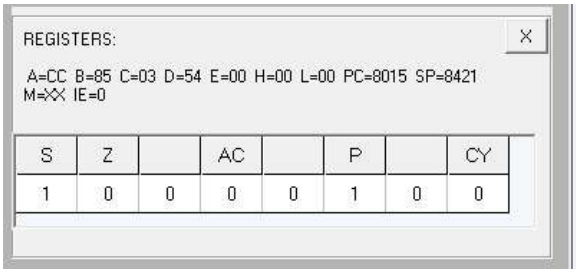


Figure 2.4.9 Values of Registers

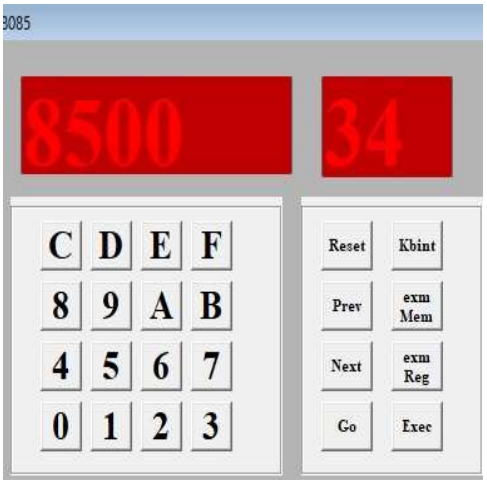


Figure 2.4.10 [8500] - 34

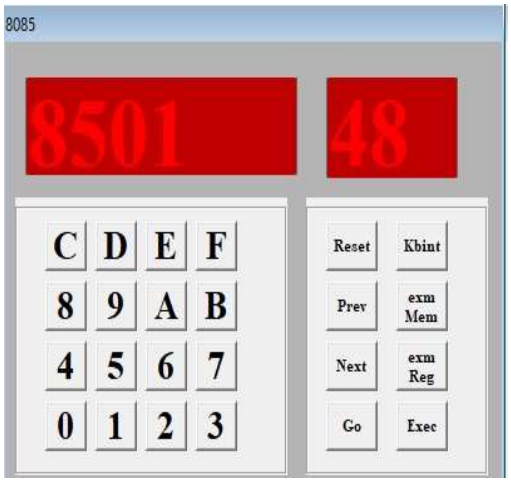


Figure 2.4.11 [8501] - 48

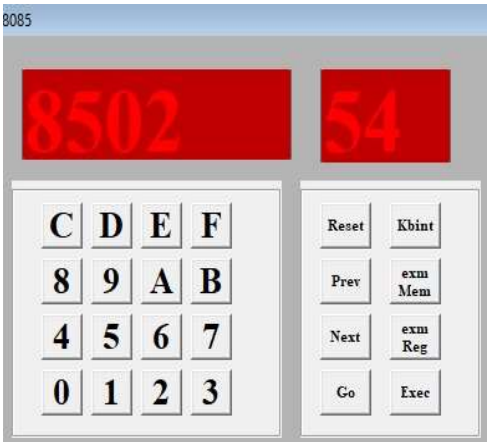


Figure 2.4.12 [8502] - 88

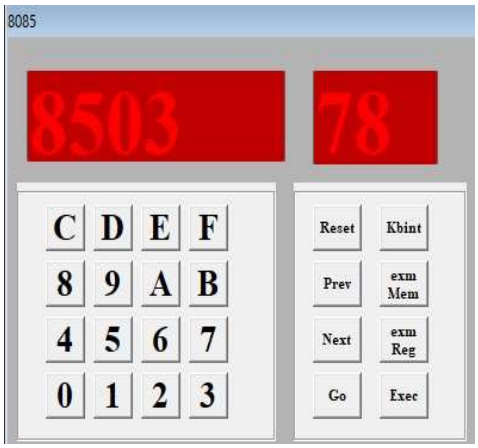


Figure 2.4.11 [8503] - 88

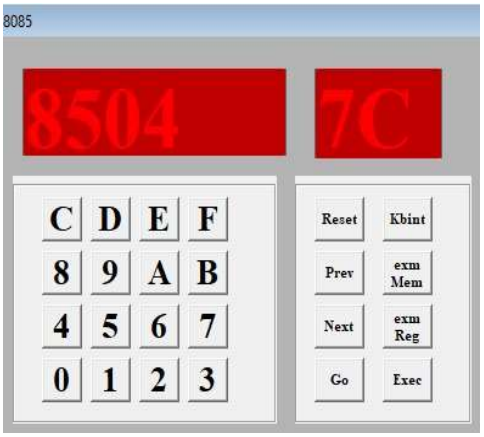


Figure 2.4.12 [8504] - 88

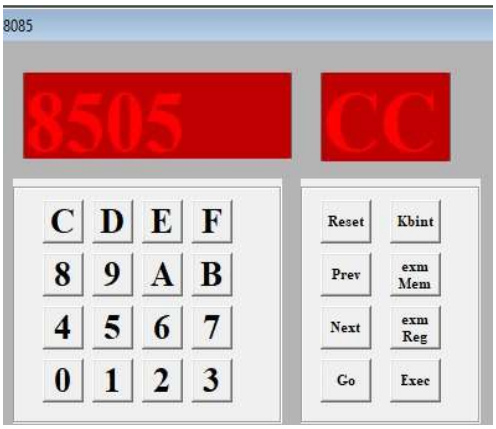


Figure 2.4.13 [8505] - 88

Input

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

Output

Registers: [D] – 48, [E] – 48, [H] – 90, [L] - 90

Memory Address: [8504] – 7C, [8505] - CC

Flags: S – 1, Z – 0, AC – 0, P – 1, CY – 0

Program No. 2.5

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

Code	Memory Location	Opcode
MVI C,00	8000, 8001	CE, 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
INX H	800C	23
MOV M,A	800D	77
INX H	800E	23
MOV M,C	800F	71
RST 5	8010	EF

Table 2.5 program to add 8-bit numbers using carry (using JNC instruction).

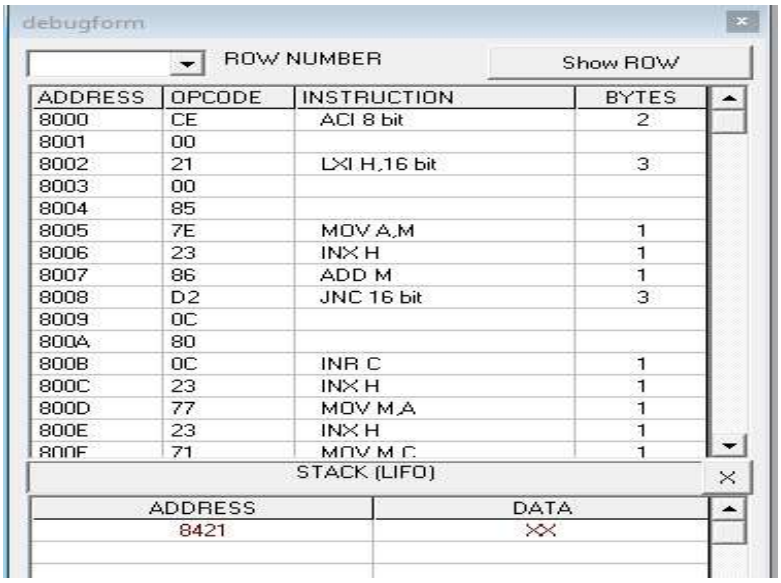


Figure 2.5.18-bit numbers using carry (using JNC instruction).

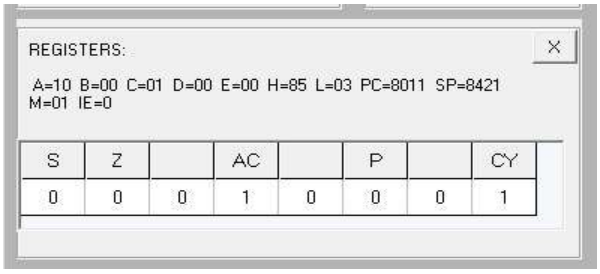


Figure 2.5.2 Values of Registers

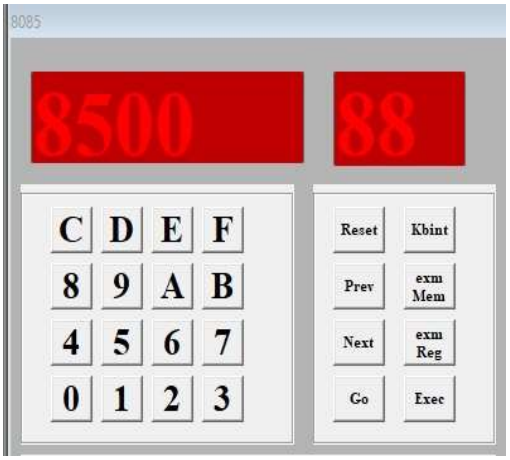


Figure 2.5. 3[8500] - 88

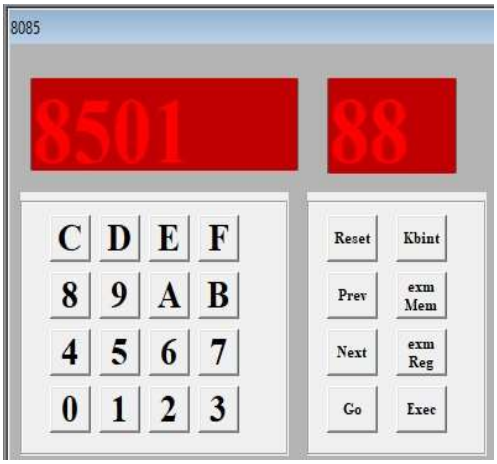


Figure 2.5. 4 [8501] - 88

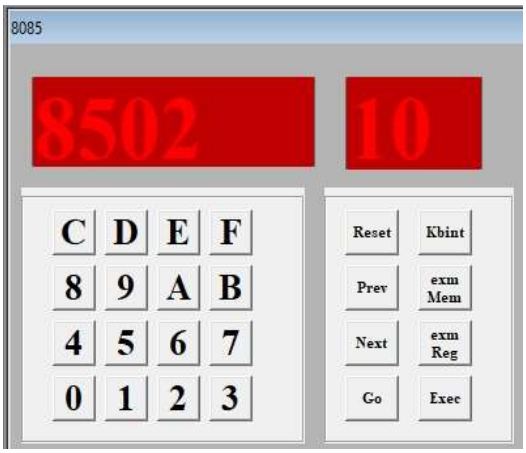


Figure 2.5.5 [8502] - 88

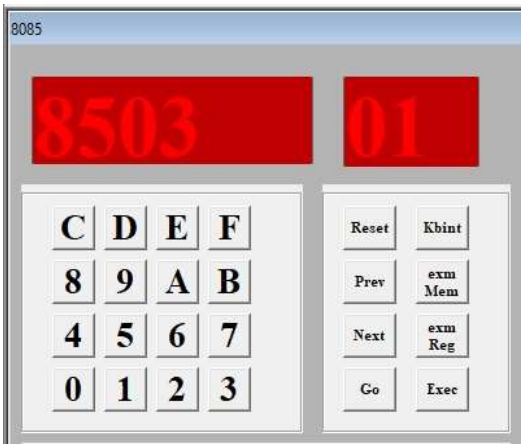


Figure 2.5.6 [8503] - 88

Input

[8500] – 88, [8501] – 88

Output

Registers: [A] – 10, [C] - 01, [F] - 11, [H] - 85, [L] - 03

Memory Address: [8502] – 10, [8503] – 01

Flags: S – 0, Z – 0, AC – 1, P – 0, CY – 1

Program No. 2.6

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

A. 1's complement:

Code	Memory Location	Opcode
LDA 8500H	8000, 8001, 8002	3A, 00, 85
CMA	8003	2F
STA 8501 H	8004, 8005, 8006	32, 01, 85
RST 5	8007	EF

Table 2.6.1 program to find 1's complement of an 8-bit number.

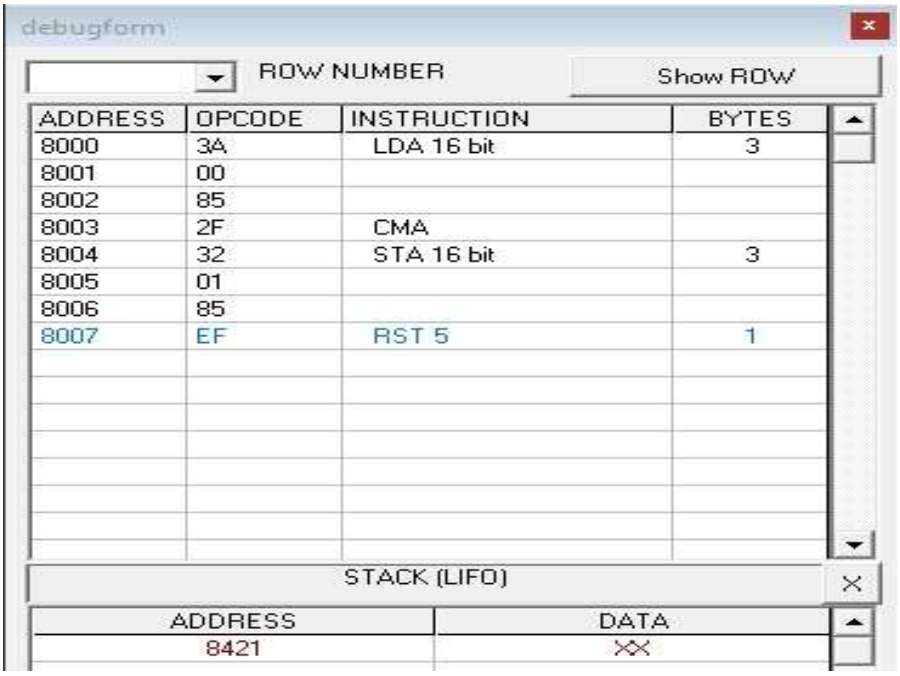


Figure 2.6.1 find 1's complement of an 8-bit number.

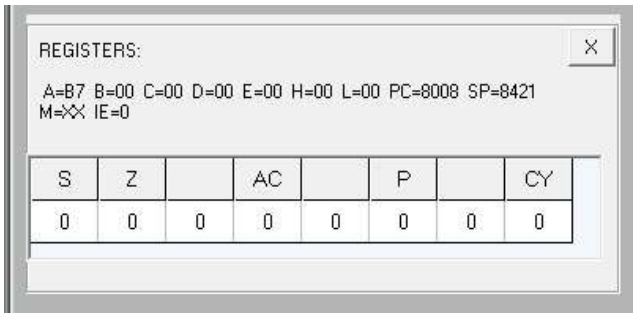


Figure 2.6.2 Values of Registers

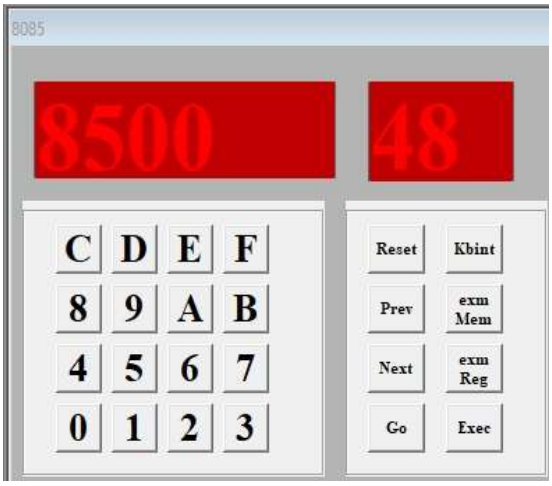


Figure 2.6.3 [8500] - 48

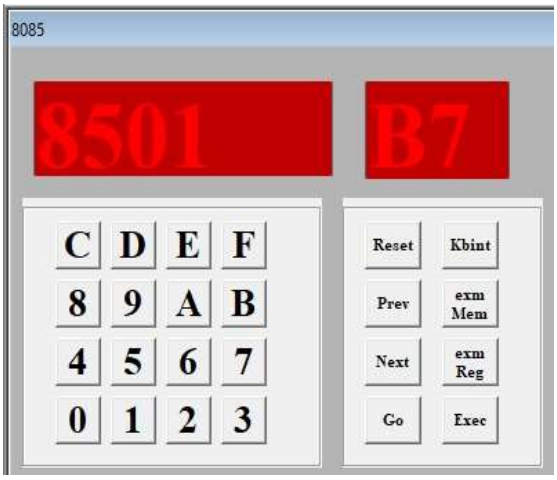


Figure 2.6.4 [8500] – B7

Input

[8500] – 48

Output

Memory Address: [8501] – B7

Flags: S – 0, Z – 0, AC – 0, P – 0, CY - 0

B. 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 2.6.2 program to find 2's complement of an 8-bit number.

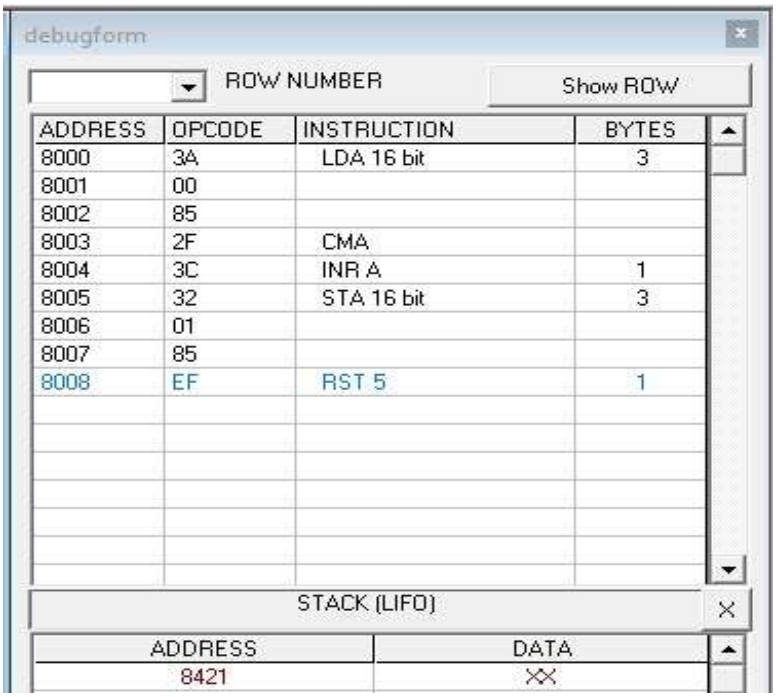


Figure 2.6.5 find 2's complement of an 8-bit number.

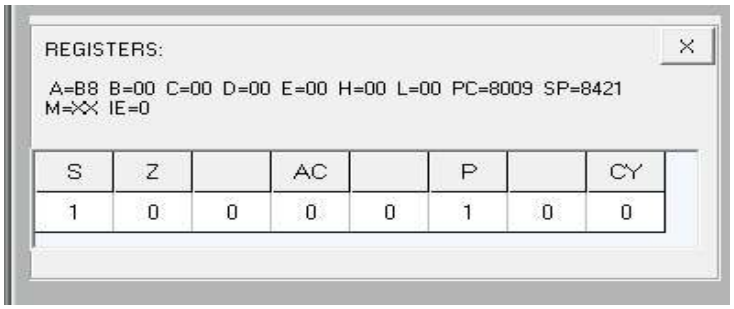


Figure 2.6.6 Values of Registers

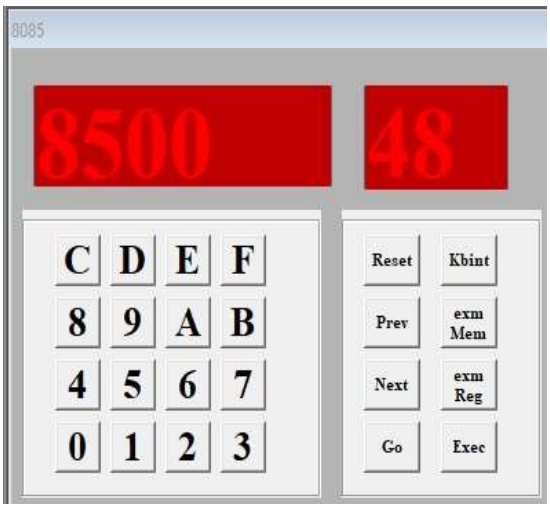


Figure 2.6.7 [8500] - 48

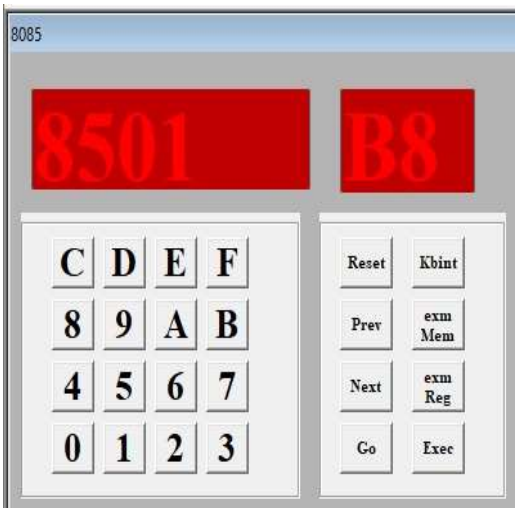


Figure 2.6. 8 [8501] – B8

Input

[8500] – 48

Output

Memory Address: [8501] – B8

Flags: S – 1, Z – 0, AC – 0, P – 1, CY - 0

Program No. 3

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

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

Table 3.1 program for the sum of series of numbers.

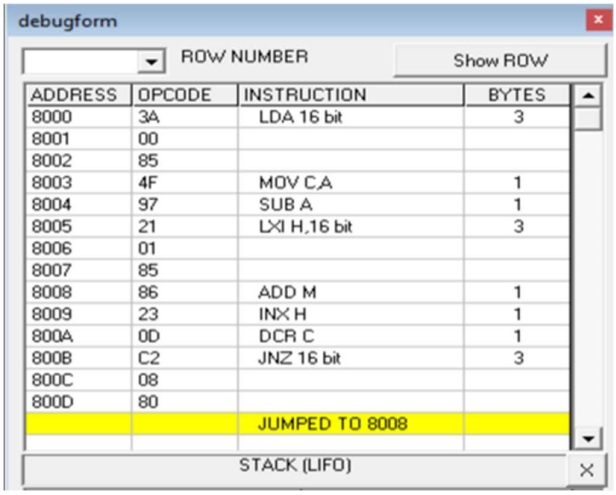


Figure 3.1 the sum of series of numbers.

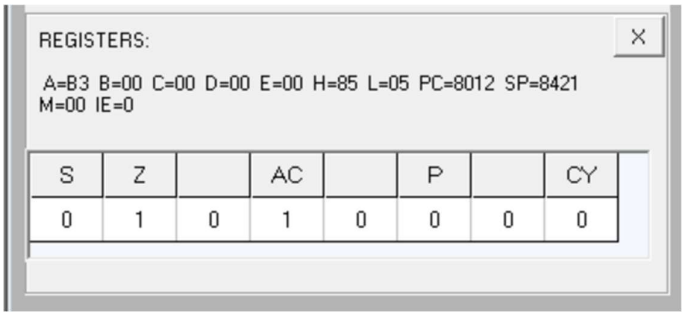


Figure 3.2 Values of Registers

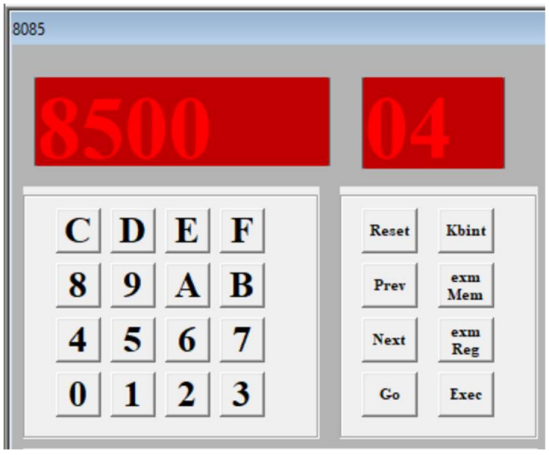


Figure 3. 3 [8500] - 04

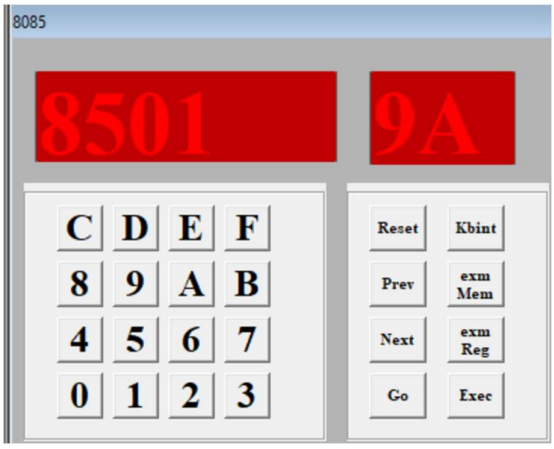


Figure 3. 4 [8501] - 88

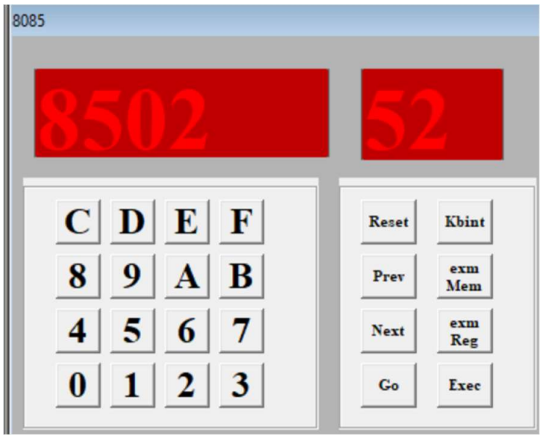


Figure 3. 5 [8502] - 88

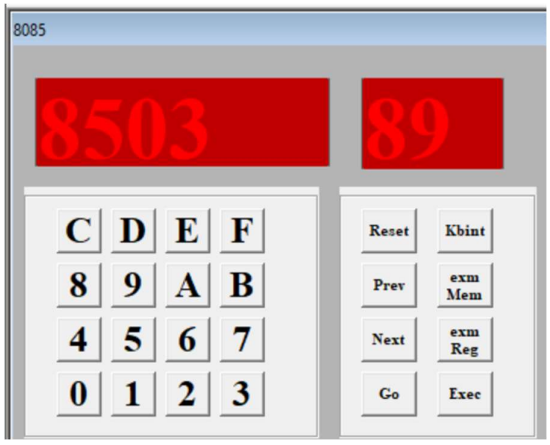


Figure 3. 6 [8503] - 88

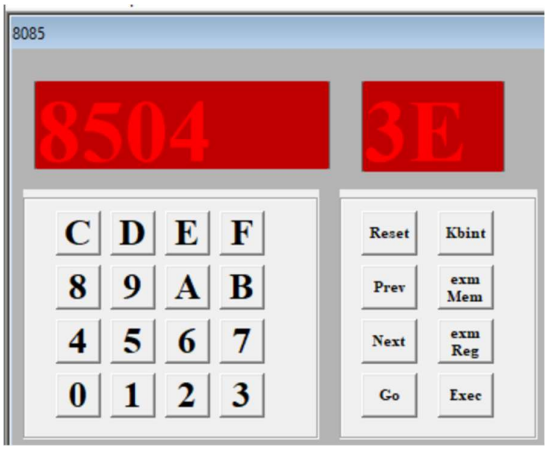


Figure 3.7 [8504] - 88

Figure 3.8 [8505] - 88.

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

Result
83

Output
Memory Address: [8600] – B3

Flags: S – 0, Z – 1, AC – 1, P – 0, CY – 0

Program 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	11,00,86
Back: MOV A, M	8008	7E
STAX D	8009	12
INX H	800A	23
INX D	800B	13
DCR C	800C	0D
JNZ Back	800D,800E,800F	C2,08,80
RST 5	8010	EF

Table 4.1 program for data transfer from memory block B1 to memory block B2.

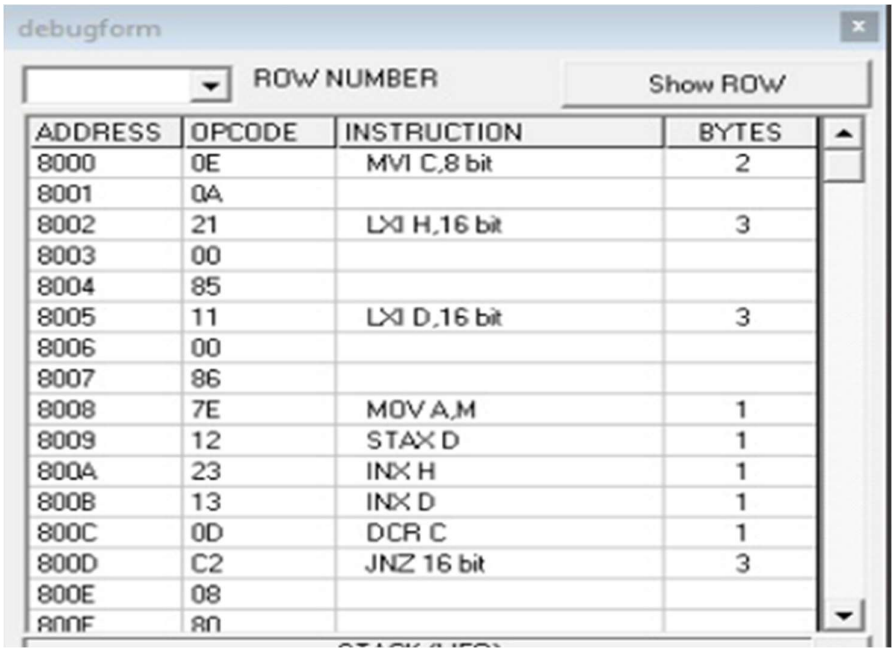


Figure 4. 1 data transfer from memory block B1 to memory block B2.

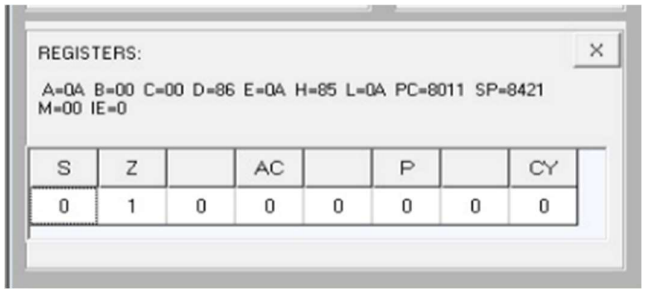


Figure 4. 2 Values of Registers

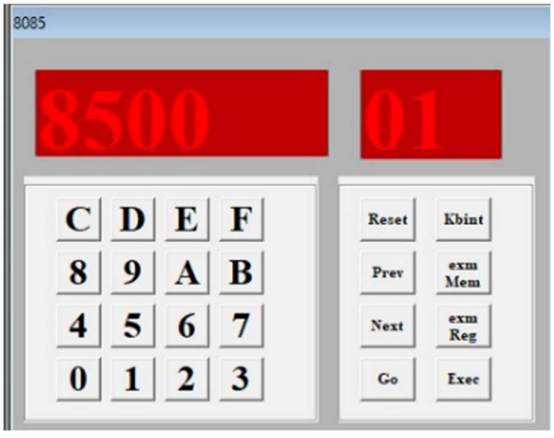


Figure 4. 3 [8500] - 01

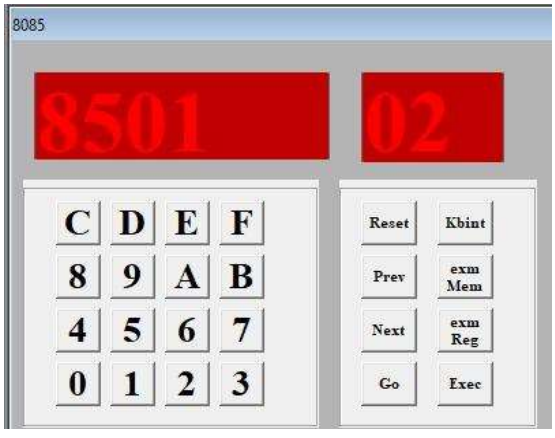


Figure 4. 4 [8501] - 02

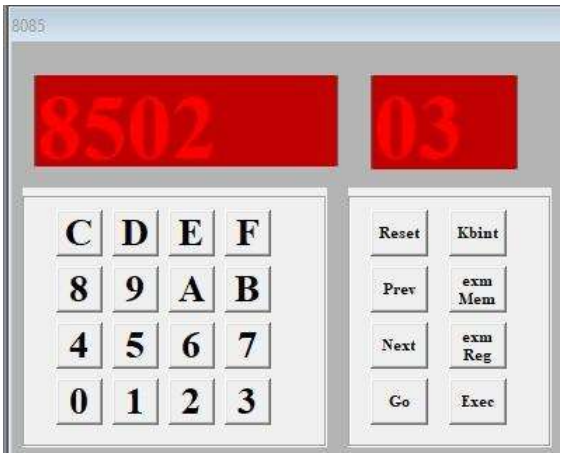


Figure 4. 5 [8502] -03

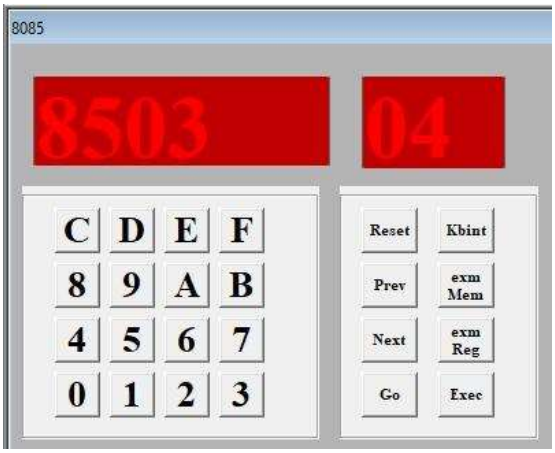


Figure 4. 6 [8503] - 04

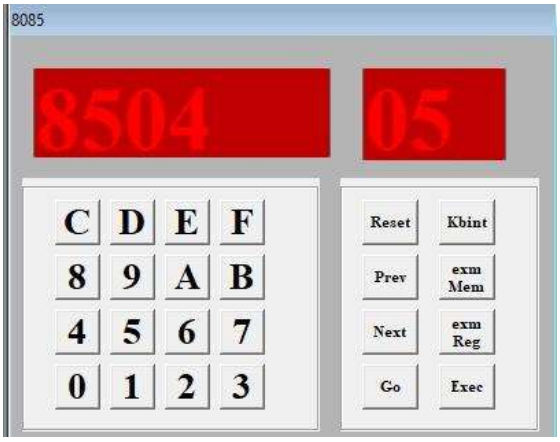


Figure 4. 7[8504] - 05

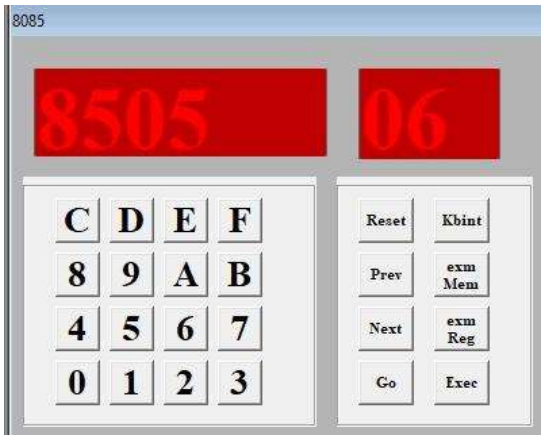


Figure 4. 8 [8505] - 06

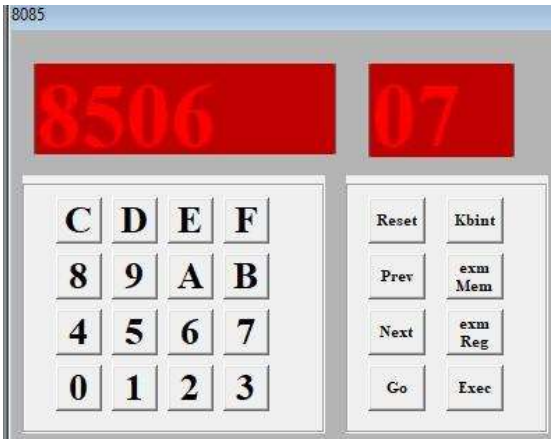


Figure 4. 9 [8506] - 07

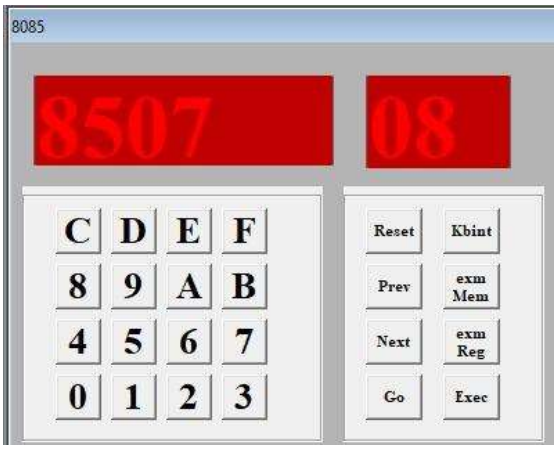


Figure 4. 10 [8507] - 08

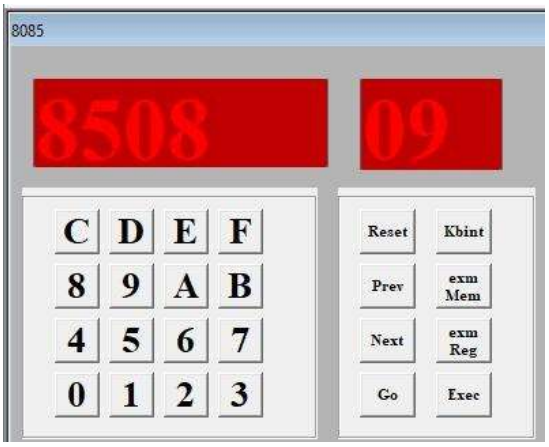


Figure 4. 11 [8508] - 09

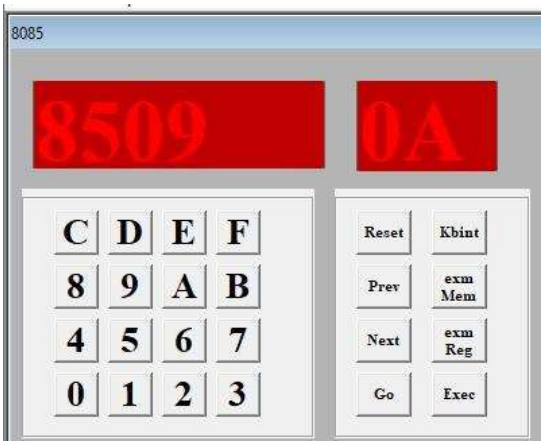


Figure 4. 12 [8509] – 0A

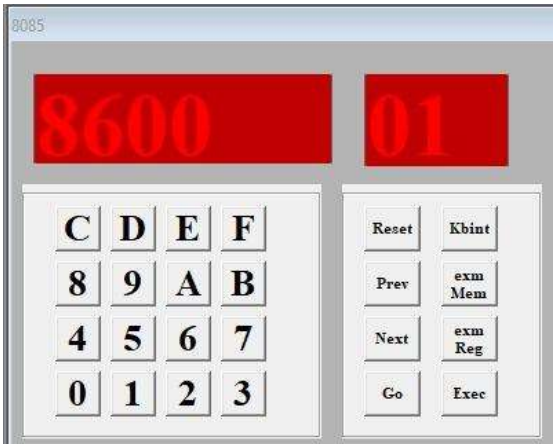


Figure 4. 13 [8600] - 01

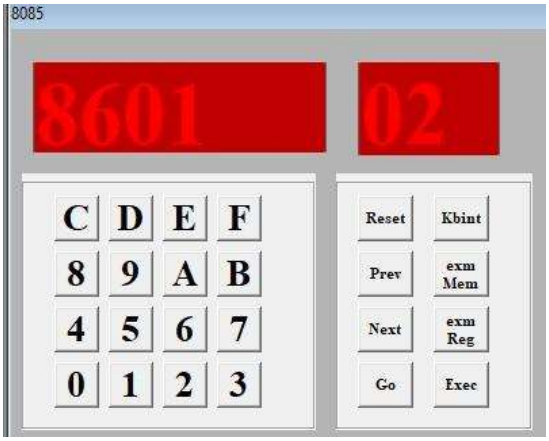


Figure 4. 14 [8601] - 02

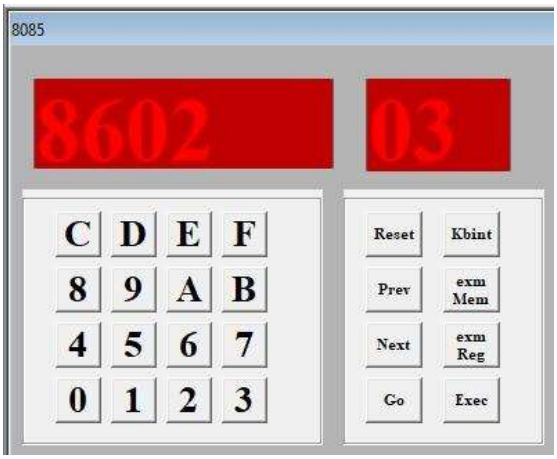


Figure 4. 15 [8500] - 88

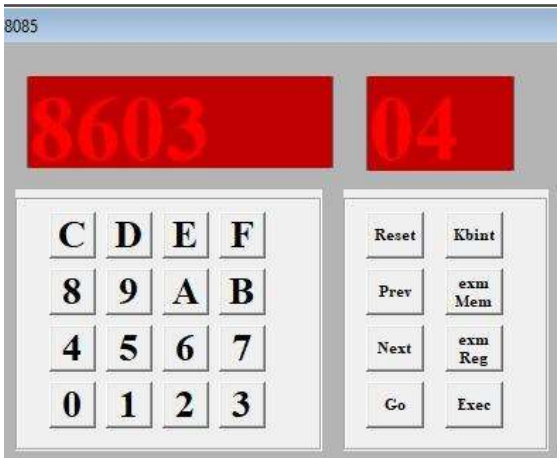


Figure 4. 16 [8500] - 88

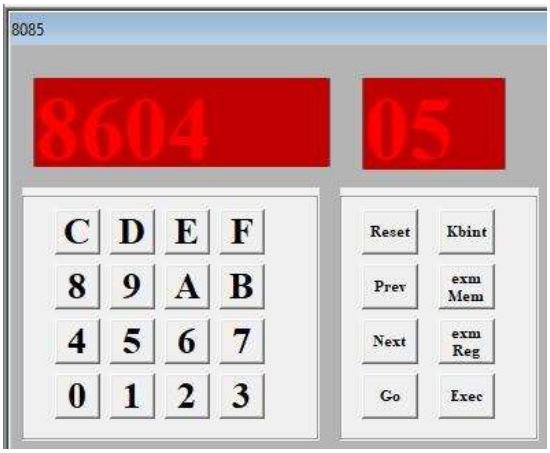


Figure 4. 17 [8604] - 05

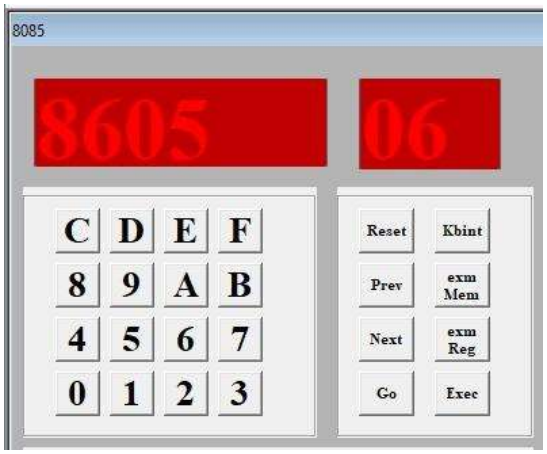


Figure 4. 18 [8605] - 06

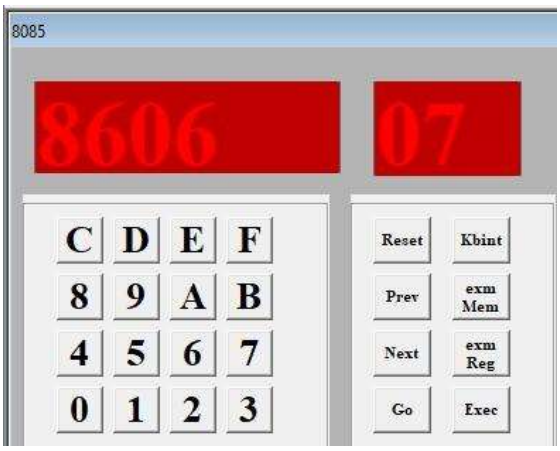


Figure 4. 19 [8606] - 07

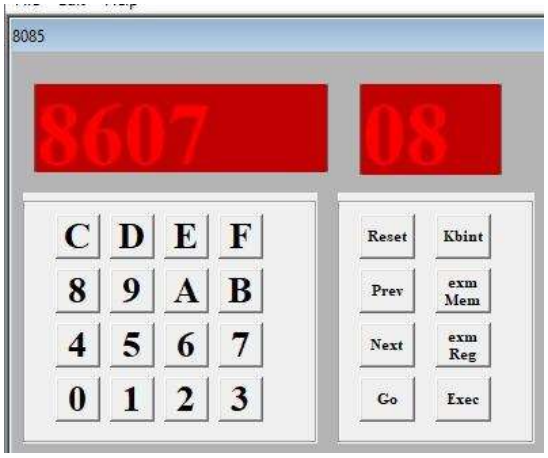


Figure 4. 20 [8607] - 08

Figure 4. 21 [8608] - 09

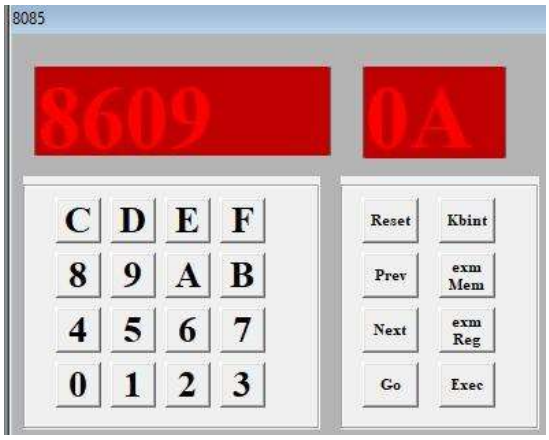


Figure 4. 22 [8609] – 0A

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

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

Program 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	8012,8013,8014	22,00,80
RST 5	8015	EF

Table 5.1 program for multiply two 8-bit numbers.

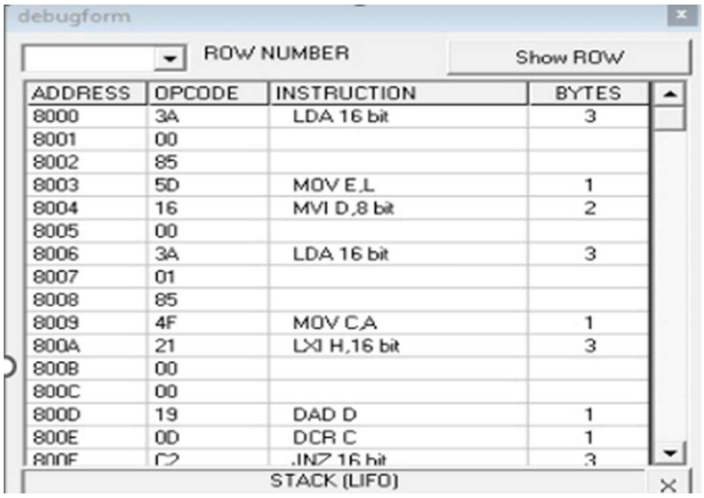


Figure 5. 1 multiply two 8-bit numbers.

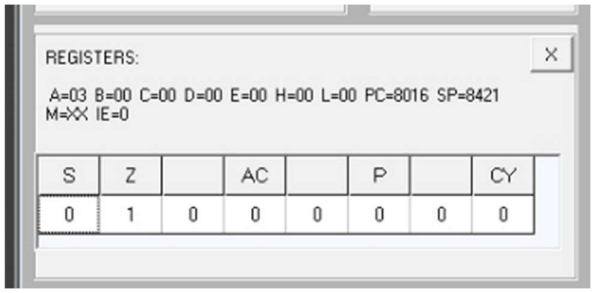


Figure 5. 2 Values of Registers

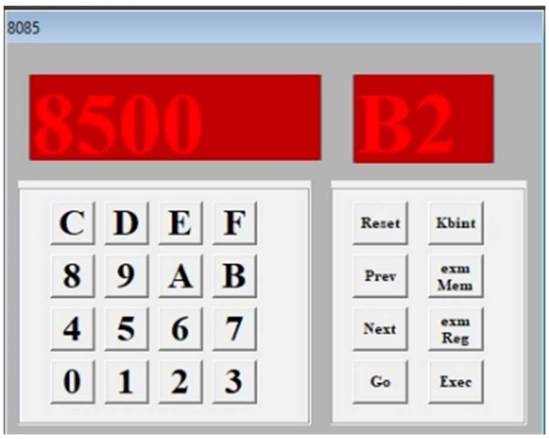


Figure 5. 3 [8500] – B2

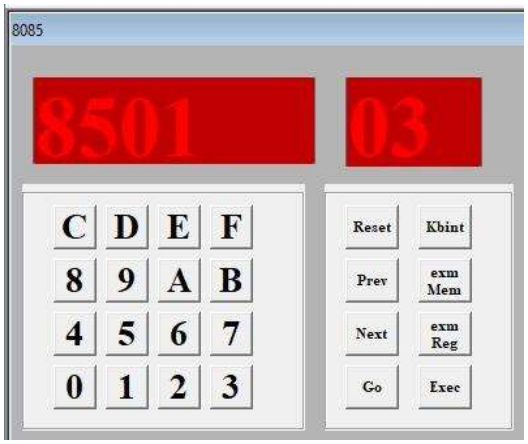


Figure 5. 4 [8501] - 03

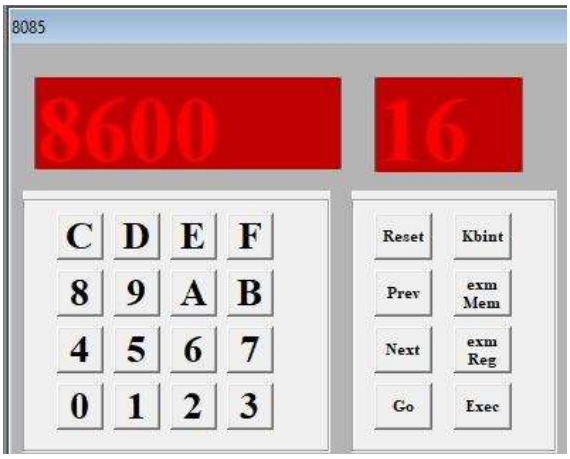


Figure 5. 5 [8600] - 16

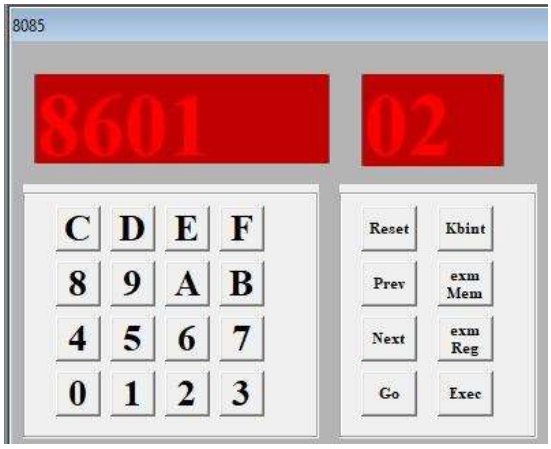


Figure 5. 6[8601] - 02

Input
[8500] – B2, [8501] – 03

Result
B2 + B2 + B2 = 0216 H

Output
Memory Address: [8600] – 16, [8601] – 02

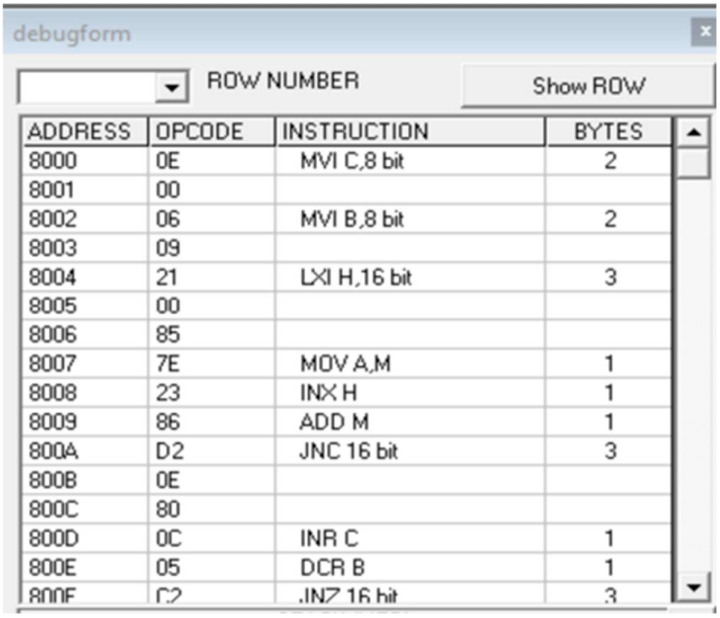
Flags: S – 0, Z – 1, AC – 0, P – 0, CY – 0

Program 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 address.

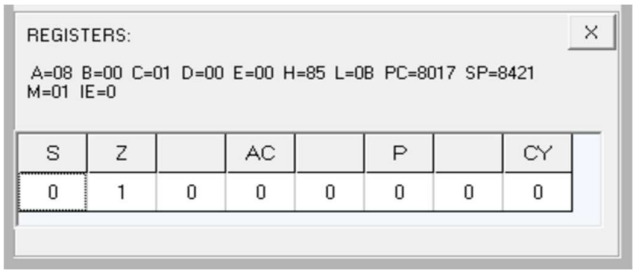
Code	Memory Location	Opcode
MVI C, 00	8000,8001	0E,00
MVI B, 09	8002,8003	06,09
LXI H, 8500H	8004,8005,8006	21,00,85
MOV A, M	8007	7C
Back: INX H	8008	23
ADD M	8009	86
JNC Next	800A,800B,800C	D2,0E,80
INR C	800D	0C
Next: DCR B	800E	05
JNZ Back	800F,8010,8011	C2,08,80
INX H	8012	23
MOV M, A	8013	77
INX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

Table 6.1 program to add ten 8-bit numbers.



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

Figure 6. 1 program to add ten 8-bit numbers.



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

Figure 6. 2 Values of Registers

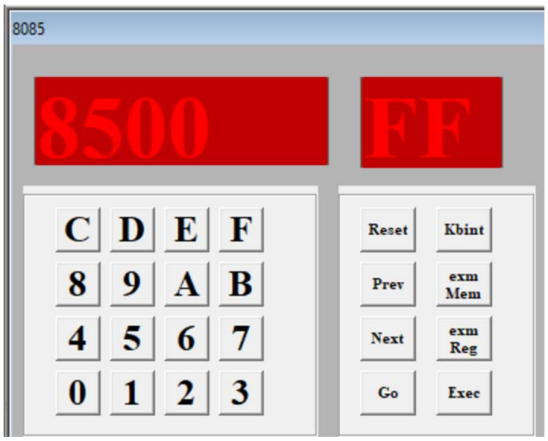


Figure 6. 3 [8500] - FF

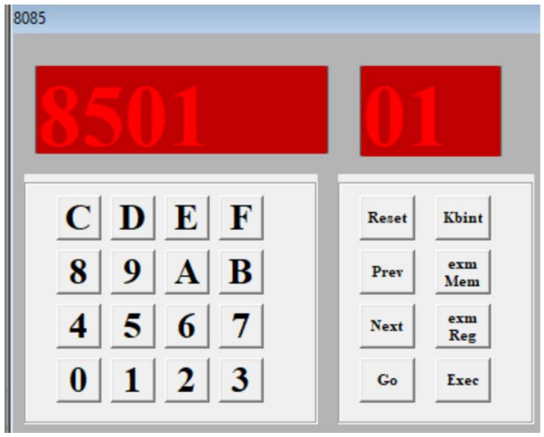


Figure 6. 4 [8501] - 01

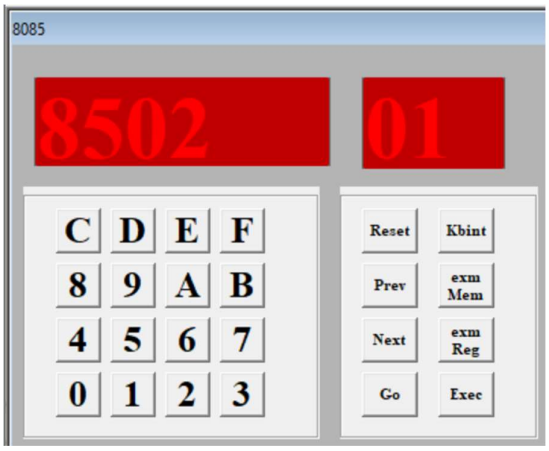


Figure 6. 5 [8502] - 01

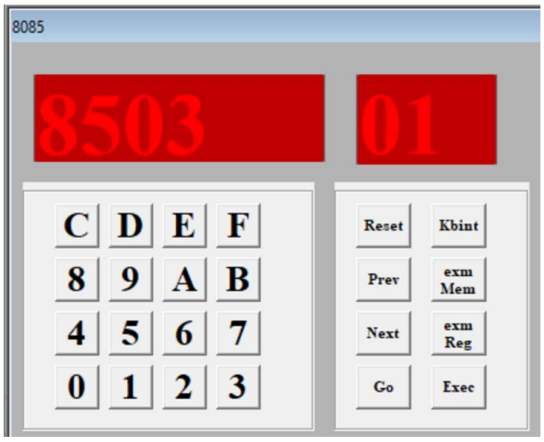


Figure 6. 6 [8503] - 01

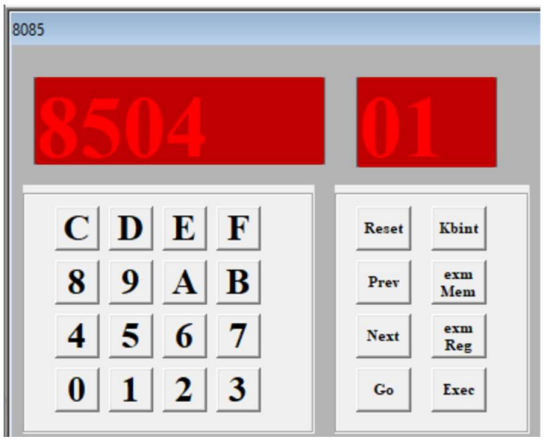


Figure 6. 7 [8504] - 01

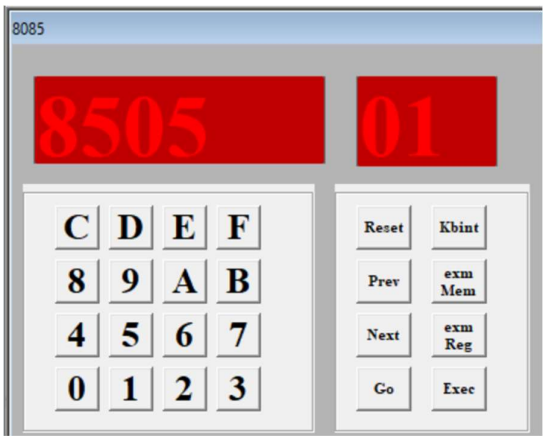


Figure 6. 8 [8505] - 01

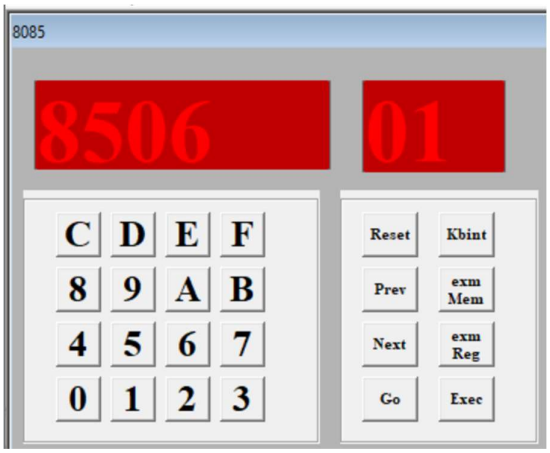


Figure 6. 9 [8506] - 01

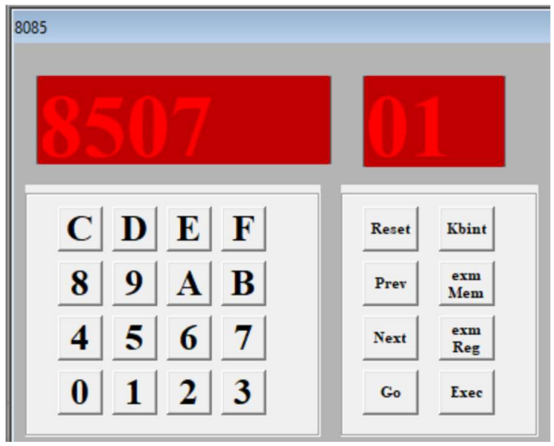


Figure 6. 10 [8507] - 01

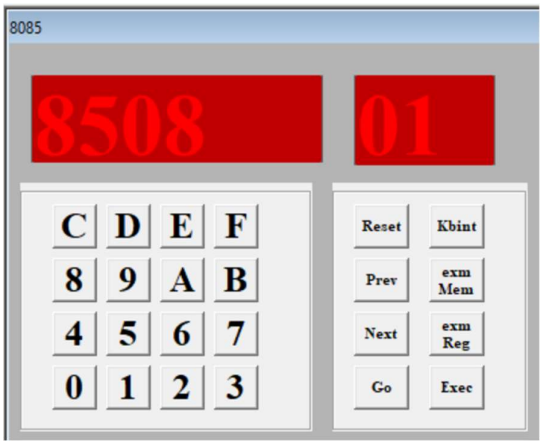


Figure 6. 11 [8508] - 01

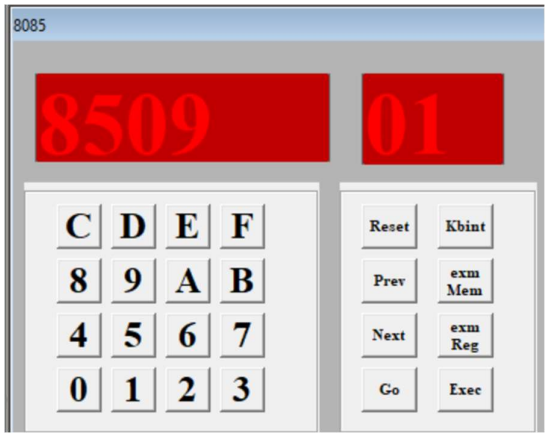


Figure 6. 12 [8509] - 01

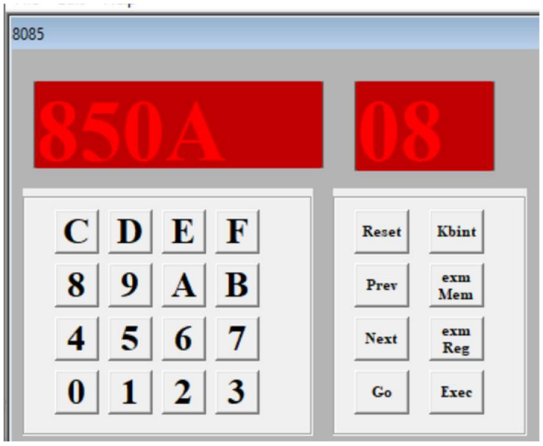


Figure 6. 13 [850A] - 08

Figure 6. 14 [850B] - 01

Input

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

Output

Memory Address: [850A] – 08, [850B] – 01

Flags: S – 0, Z – 1, AC – 0, P – 0, CY –0

Program 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 7.1 program to find the negative numbers in a block of data.

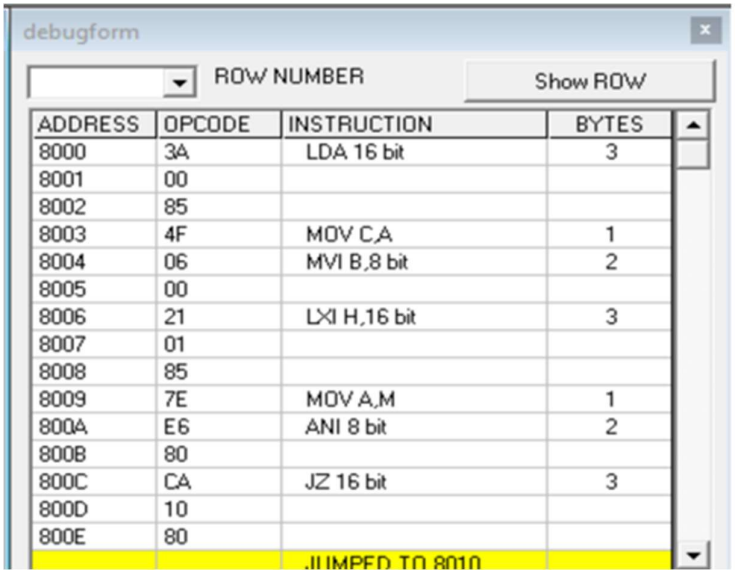


Figure 7.1 find the negative numbers in a block of data.

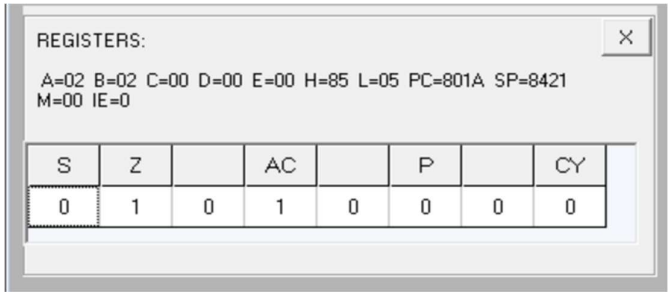


Figure 7. 2 Values of Registers

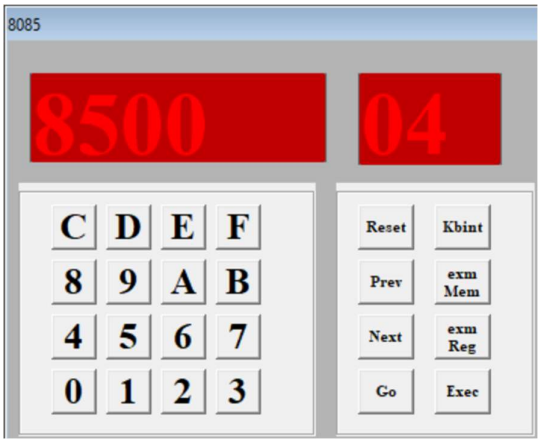


Figure 7. 3 [8500] - 04

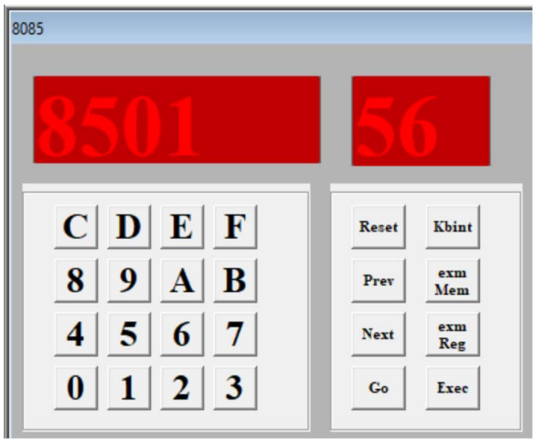


Figure 7. 4 [8501] - 56

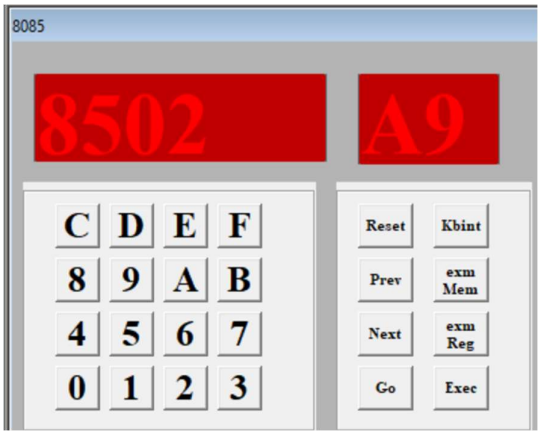


Figure 7. 5 [8502] – A9

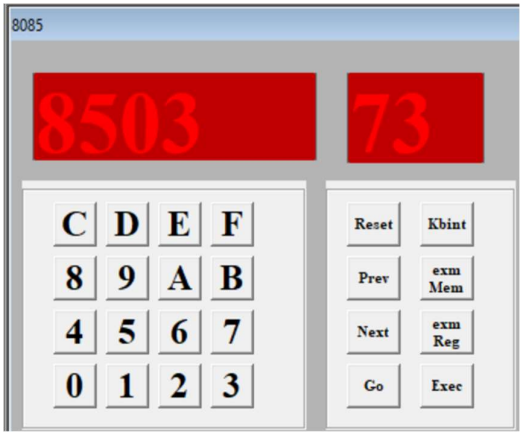


Figure 7. 6 [8503] - 73

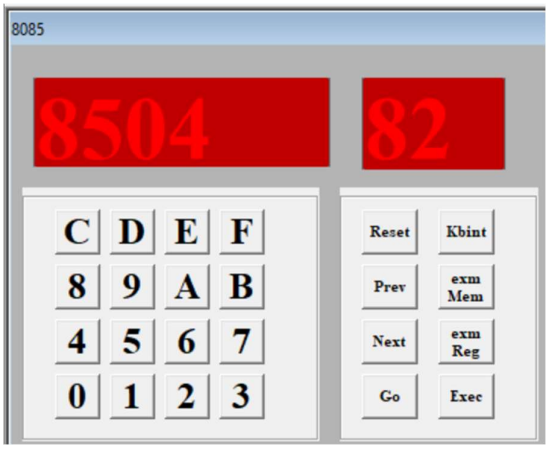


Figure 7. 7 [8504] - 82

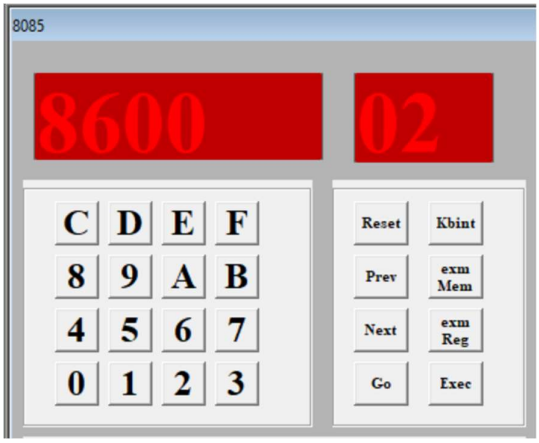


Figure 7. 8 [8600] - 02

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

Result
02

Output
Memory Address: [8600] – 02

Flags: S – 0, Z – 1, AC – 1, P – 0, CY – 0

Program 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, 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

Table 8.1 program to count the number of one's in a number.

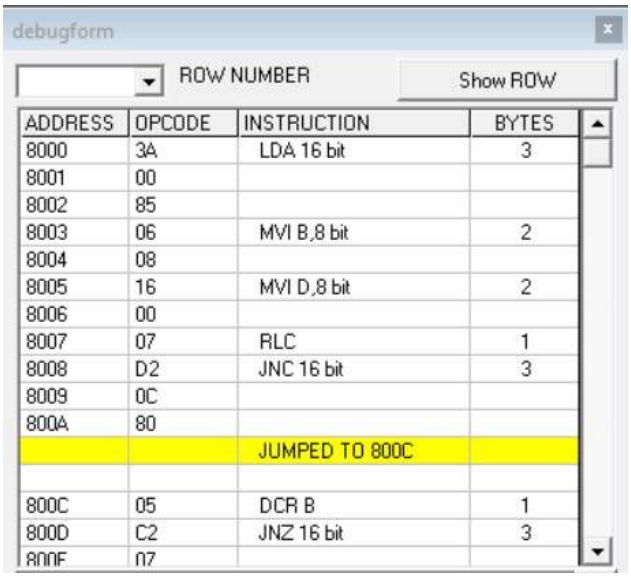


Figure 8.1 count the number of one's in a number.

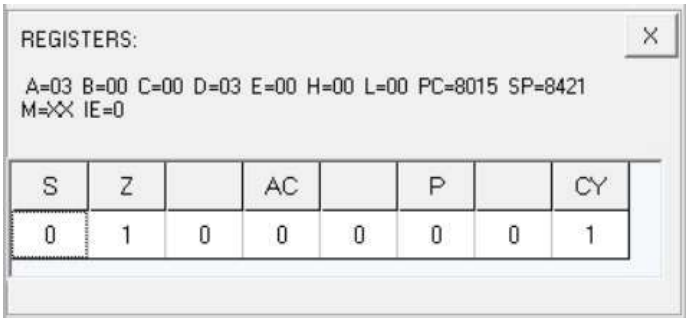


Figure 8.2 Values of Registers

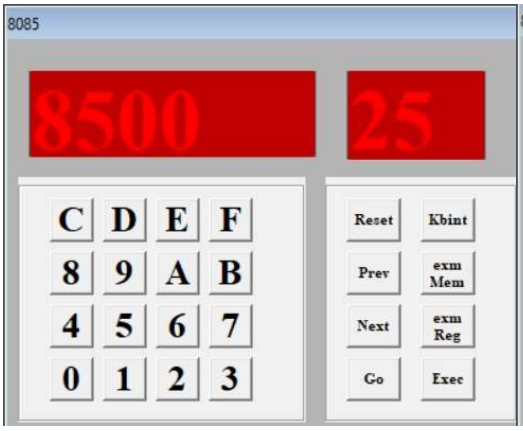


Figure 8.3 [8500] - 25

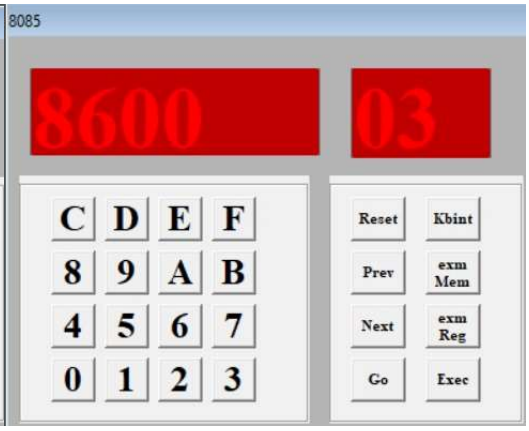


Figure 8.4 [8600] - 03

Input - [8500] – 25 0010 0101

Output - [8600] – 03

Program 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	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
RST 5	801C	EF

Table 9.1 program to arrange numbers in Ascending order.

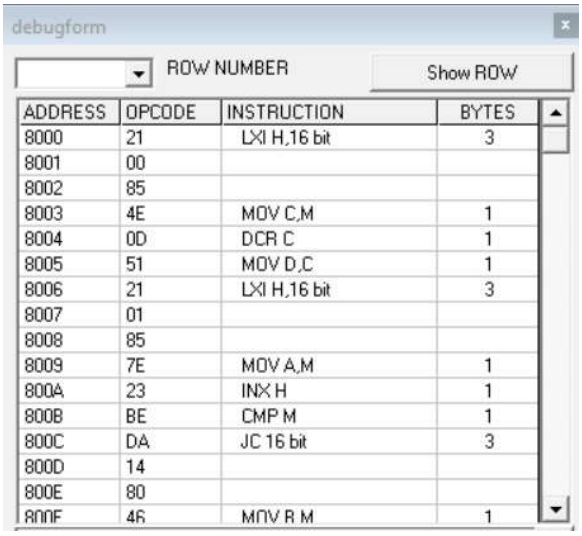


Figure 9.1 Arrange numbers in Ascending order

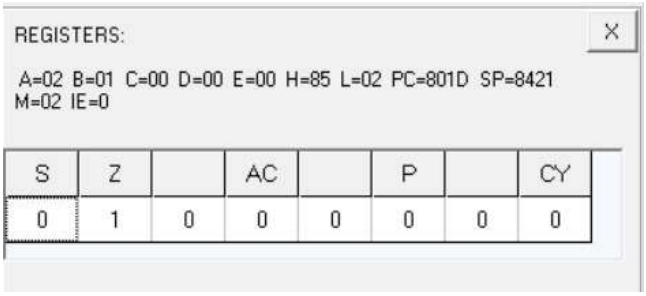


Figure 9.2 Values of Registers

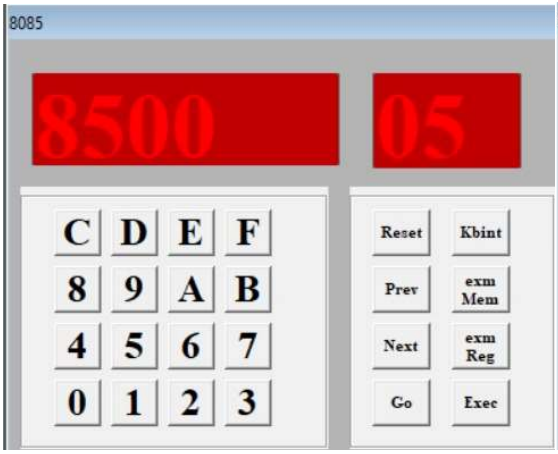


Figure 9.3 [8500] - 05

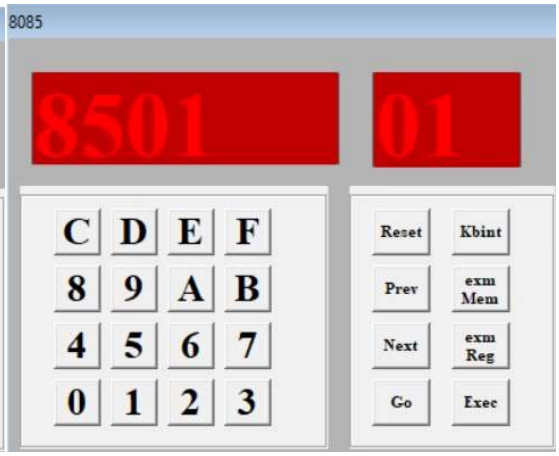


Figure 9.4 [8501] - 01

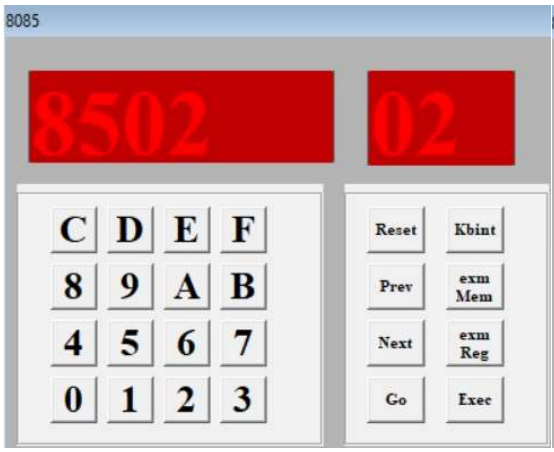


Figure 9.5 [8502] - 02

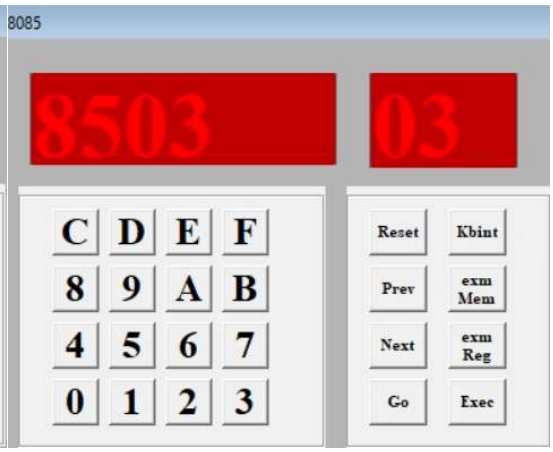


Figure 9.6 [8503] - 03

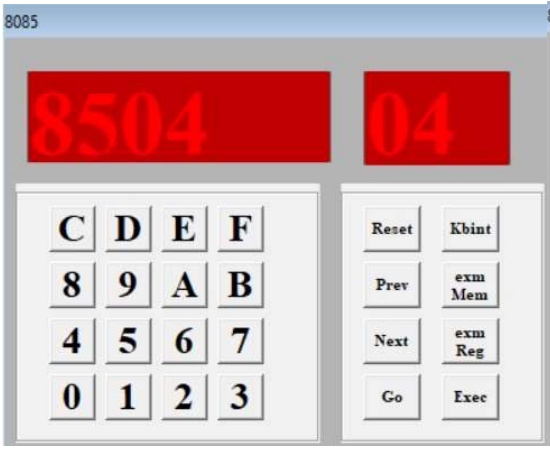


Figure 9.7 [8504] - 04

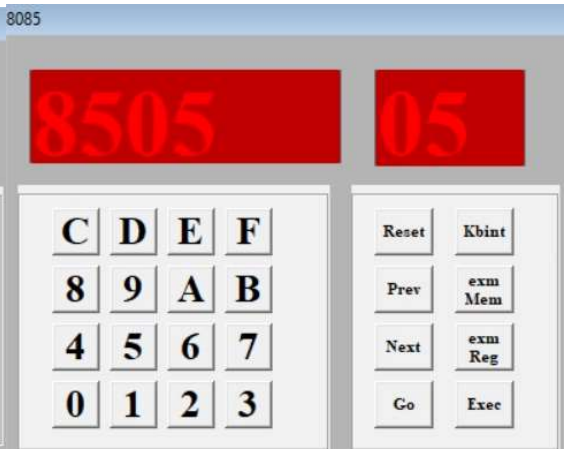


Figure 9.8 [8505] - 05

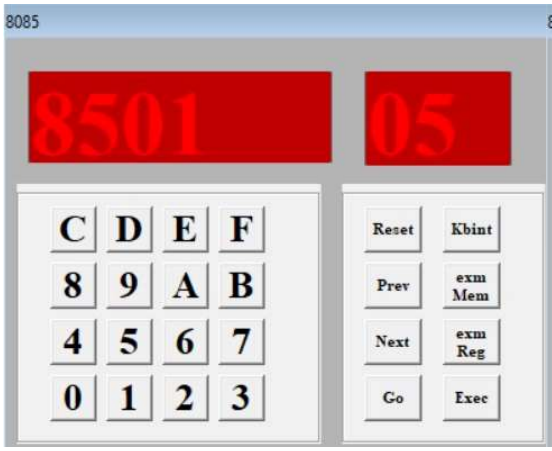


Figure 9.9 [8501] - 05

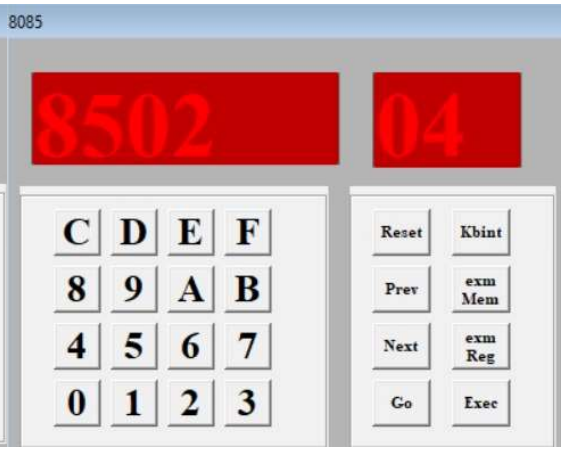


Figure 9.10 [8502] - 04

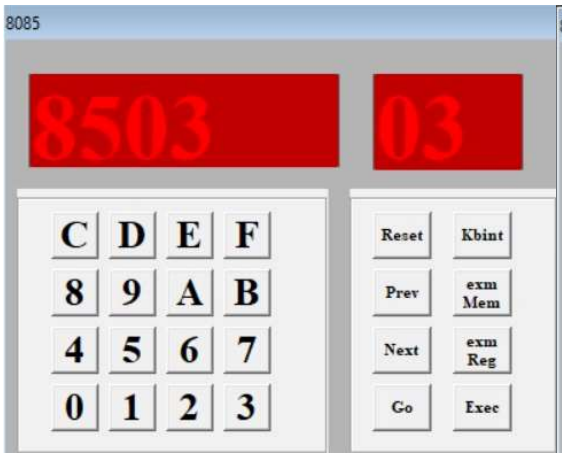


Figure 9.11 [8503] - 03

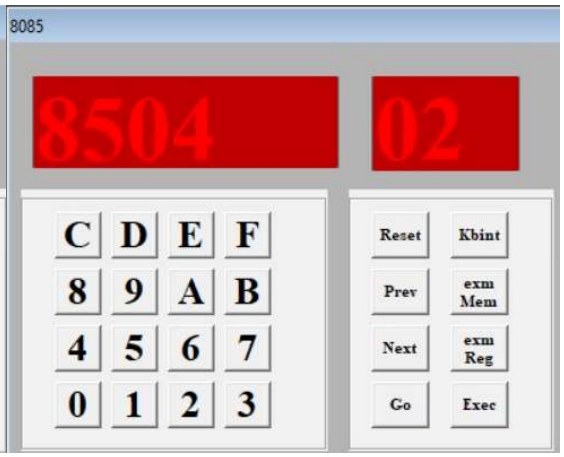


Figure 9.12 [8504] - 02

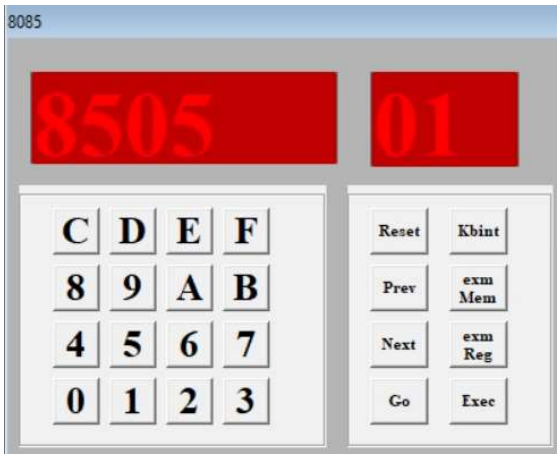


Figure 9.13 [8505] - 01

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

Program No. 10

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

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

Table 10.1 program to calculate the sum of series of even numbers.

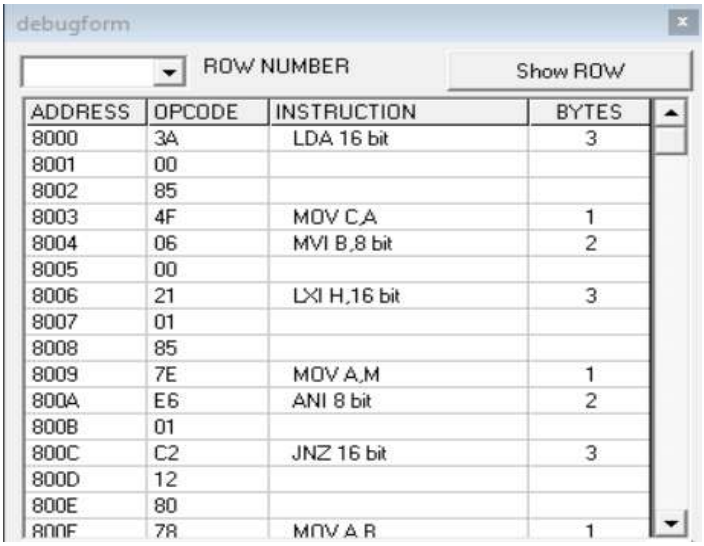


Figure 10.1 Calculate the sum of series of even numbers.

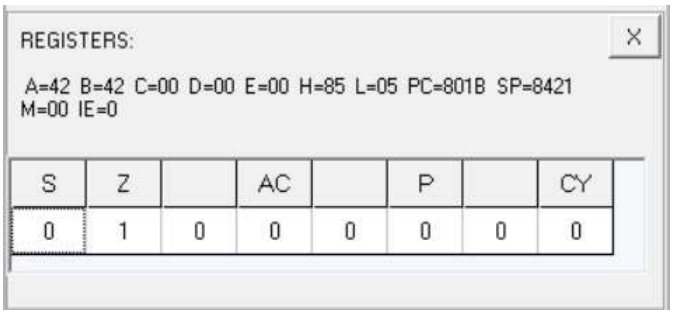


Figure 10.2 Values of Registers

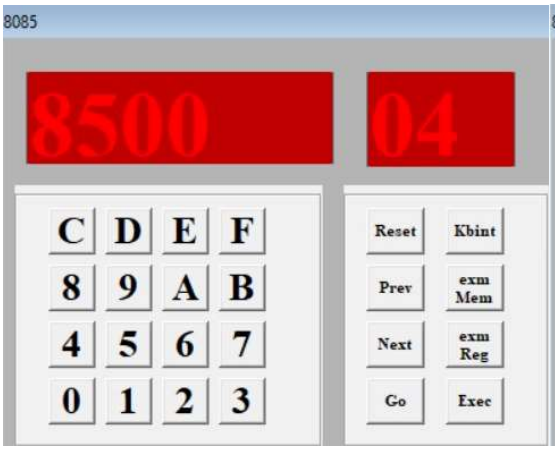


Figure 10.3 [8500] - 04

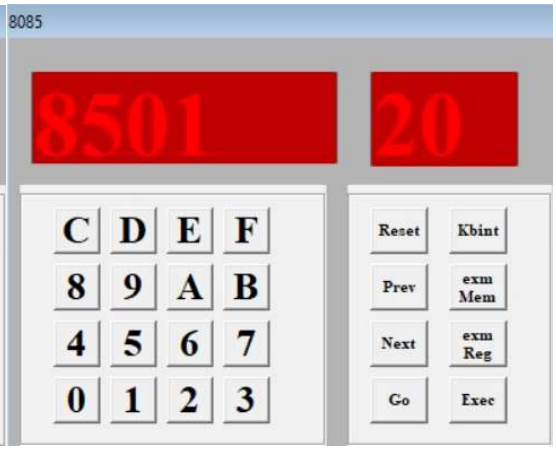


Figure 10.4 [8501] - 20

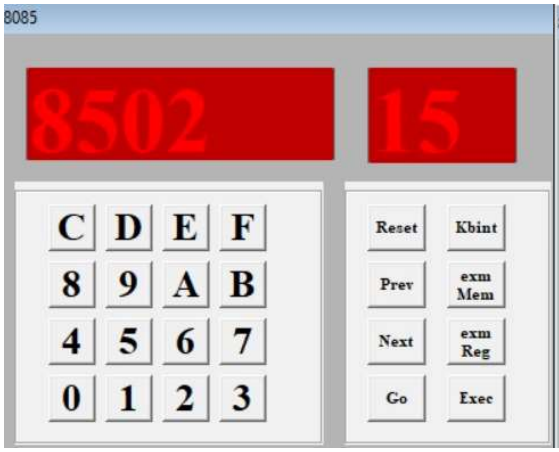


Figure 10.5 [8502] - 15

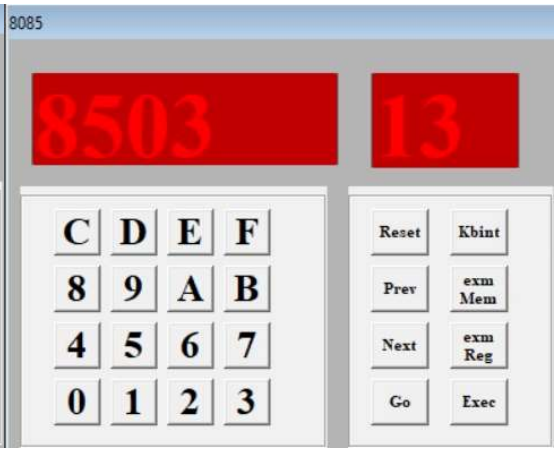


Figure 10.6 [8503] - 13

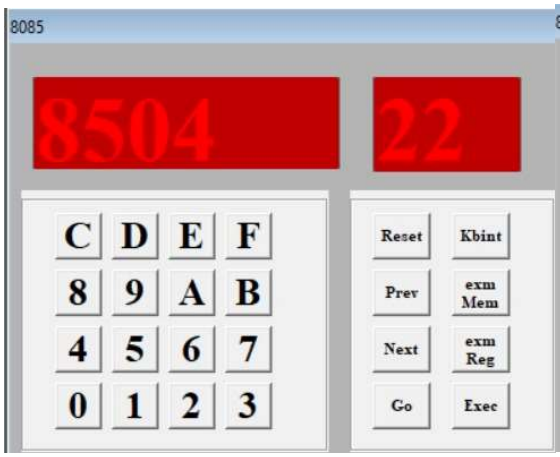


Figure 10.7 [8504] - 22

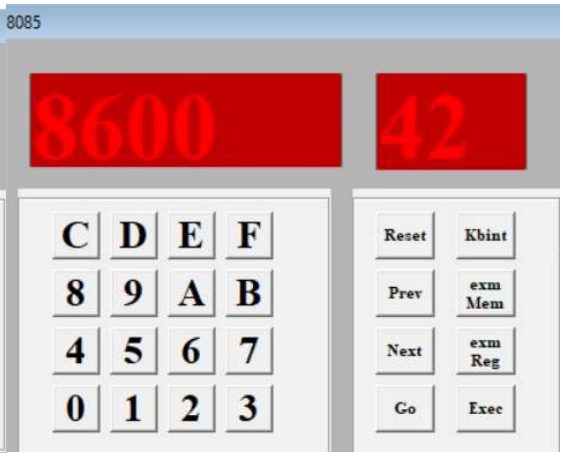


Figure 10.8 [8600] - 42

Input

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

Output

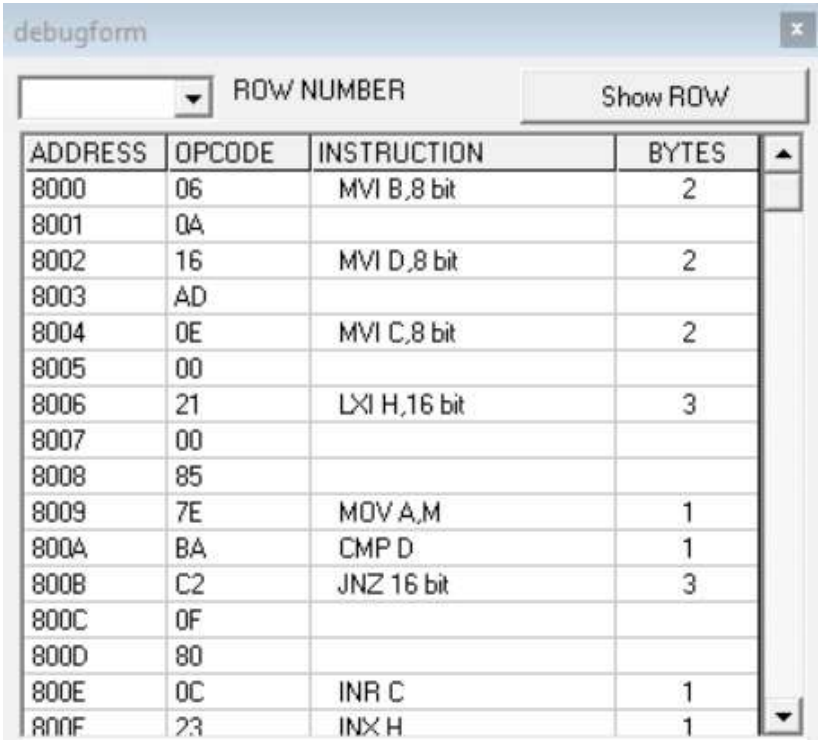
[8600] – 42

Program No. 11

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

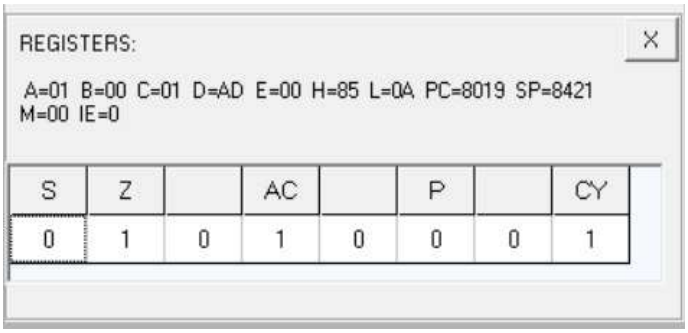
Code	Memory Location	Opcode
MVI B, 0A	8000, 8001	06, 0A
MVI D, AD	8002, 8003	16, AD
MVI C, 00	8004, 8005	0E, 00
LXI H, 8500H	8006, 8007, 8008	21, 00, 85
Back: MOV A, M	8009	7E
CMP D	800A	BA
JNZ Next	800B, 800C, 800D	C2, 0F, 80
INR C	800E	0C
Next: INX H	800F	23
DCR B	8010	05
JNZ Back	8011, 8012, 8013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

Table 11.1 Assembly language program to verify how many bytes are present in a given set.



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

Figure 11.1 to verify how many bytes are present in a given set



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

Figure 11.2 Values of Registers

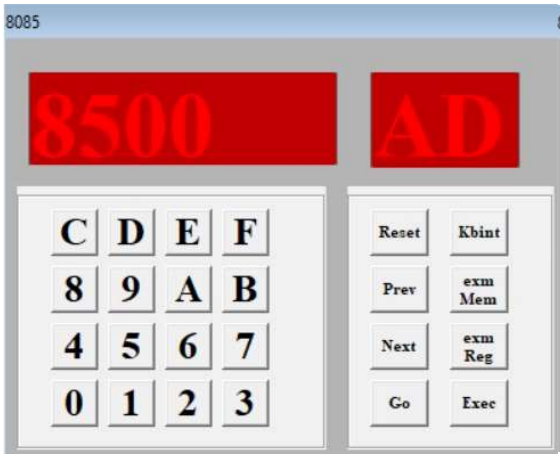


Figure 11.3 [8500] - AD

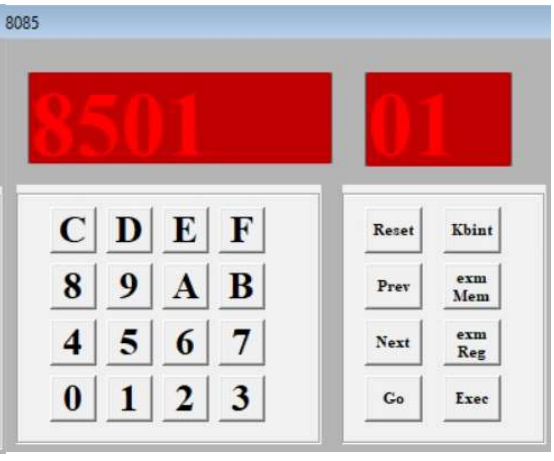


Figure 11.4 [8501] - 01

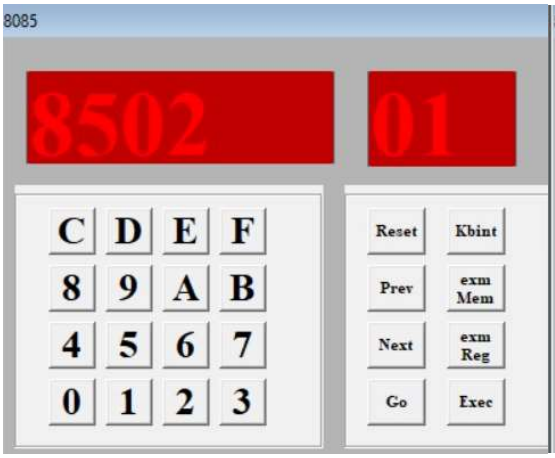


Figure 11.5 [8502] – 01

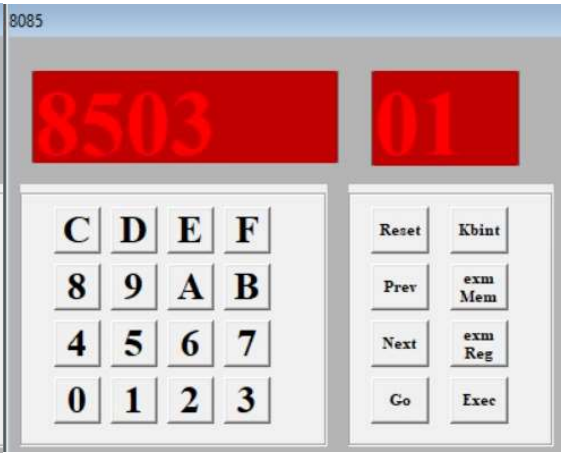


Figure 11.6 [8503] - 01

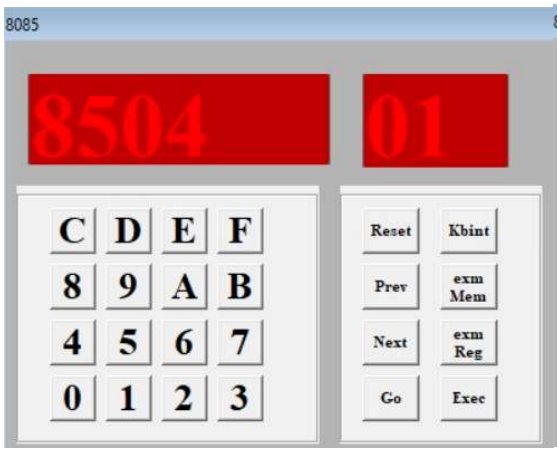


Figure 11.7 [8504] - 01

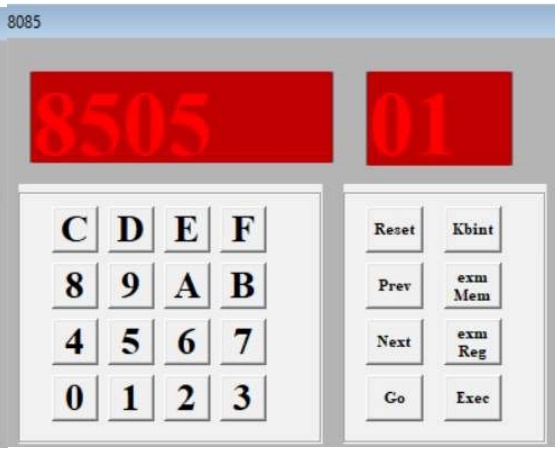


Figure 11.8 [8505] - 01

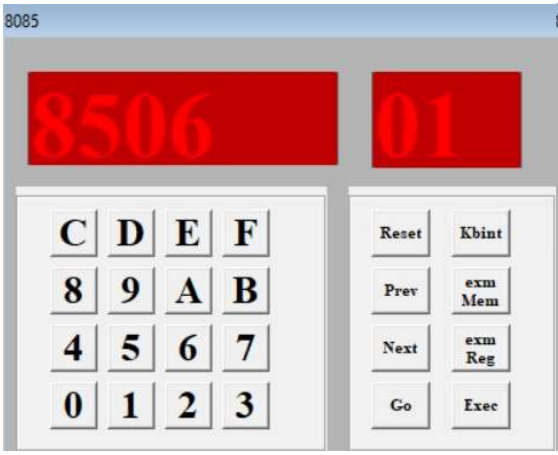


Figure 11.9 [8506] - 01

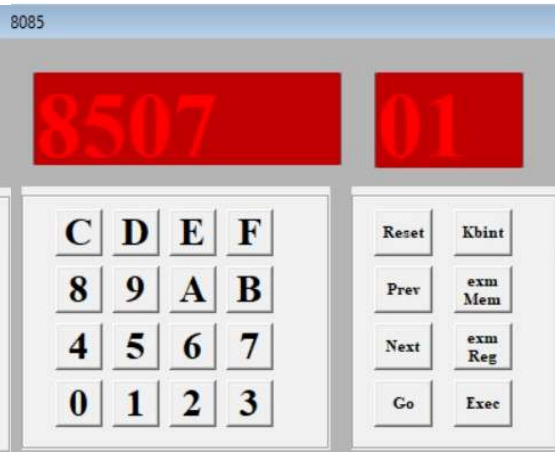


Figure 11.10 [8507] - 01

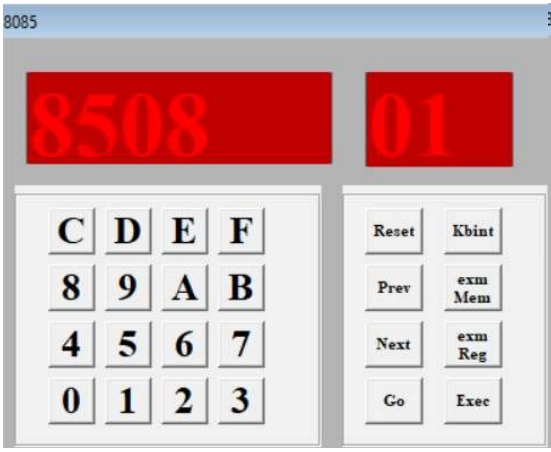


Figure 11.11 [8508] - 01

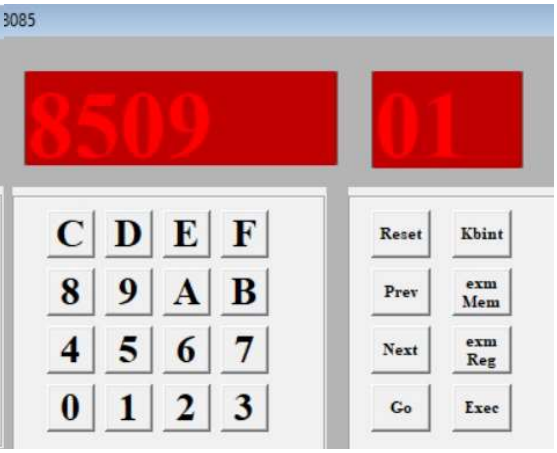


Figure 11.12 [8509] - 01

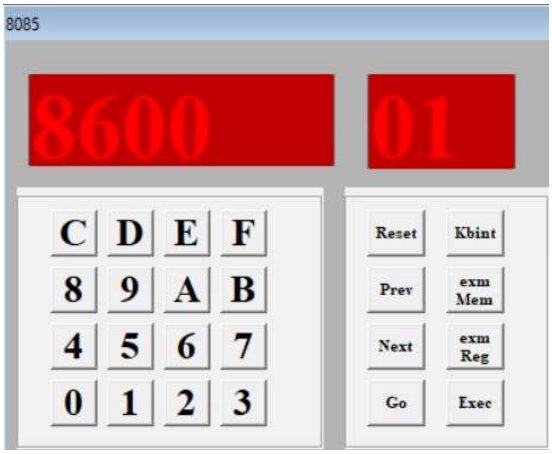


Figure 11.13 [8600] - 01

Input

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

Output

[8600] – 01

Program No. 12

Aim: Assembly language program to find the numbers of even parity in ten consecutive memory locations in 8085.

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

Table 12.1 Assembly language program to find the numbers of even parity in ten consecutive memory locations in 8085.

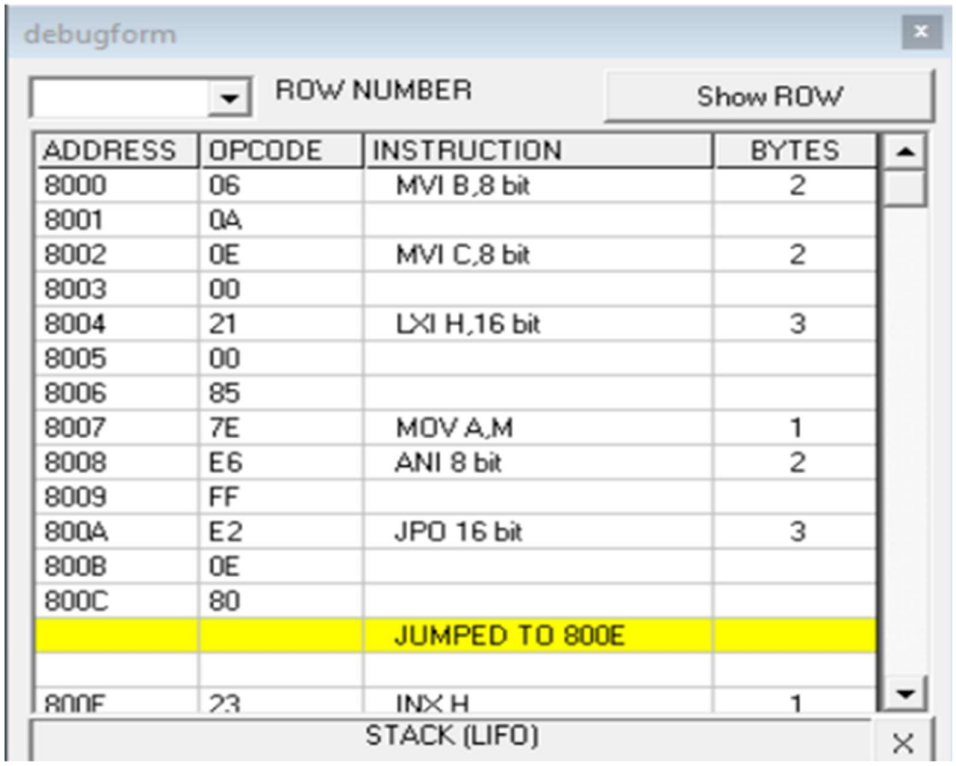


Figure 12.1 find the numbers of even parity in ten consecutive memory locations

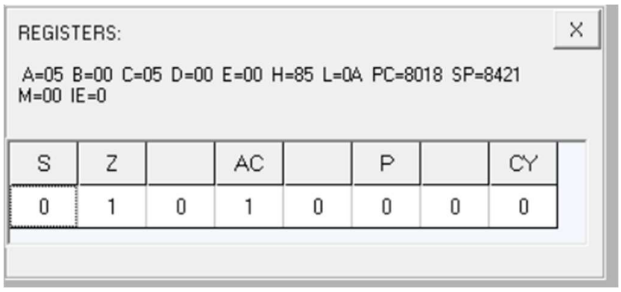


Figure 12.2 Values of Registers

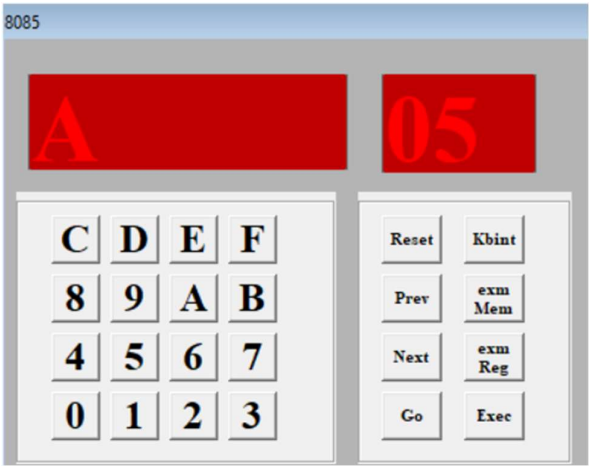


Figure 12.3 [A] - 05

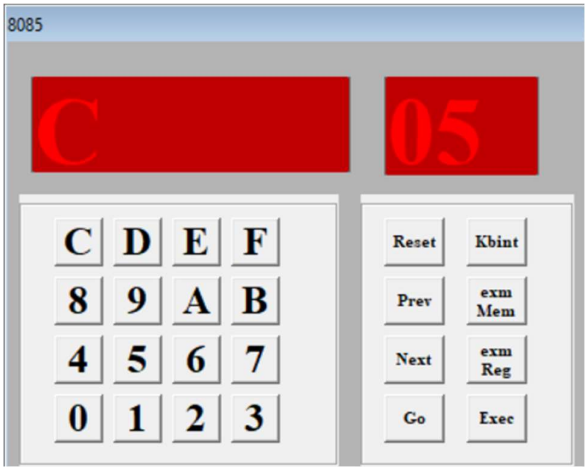


Figure 12.4 [C] - 05

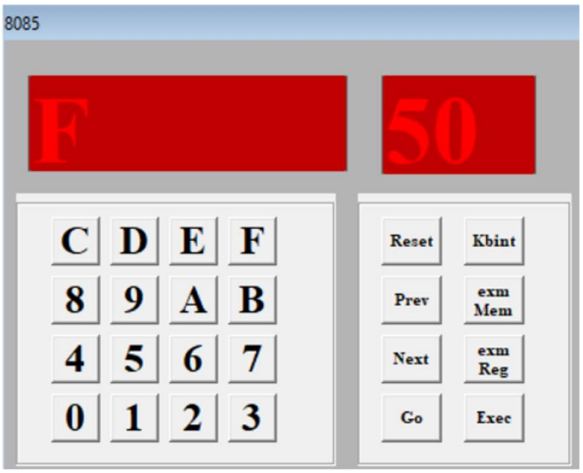


Figure 12.5 [F] - 50

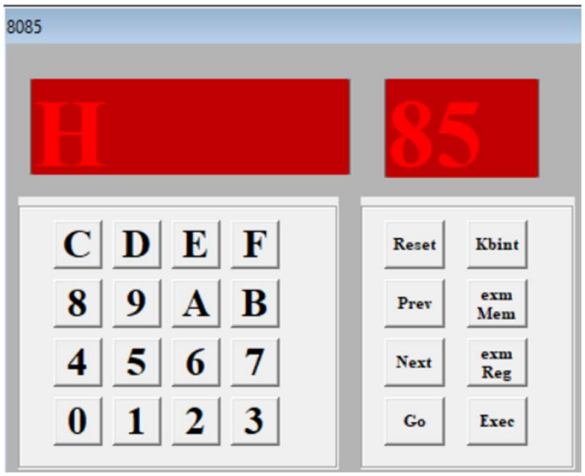


Figure 12.6 [H] - 85

Figure 12.7 [L] – 0A

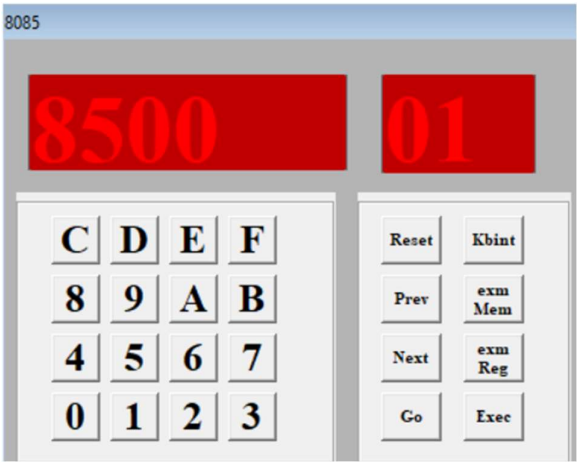


Figure 12.8 [8500] - 01

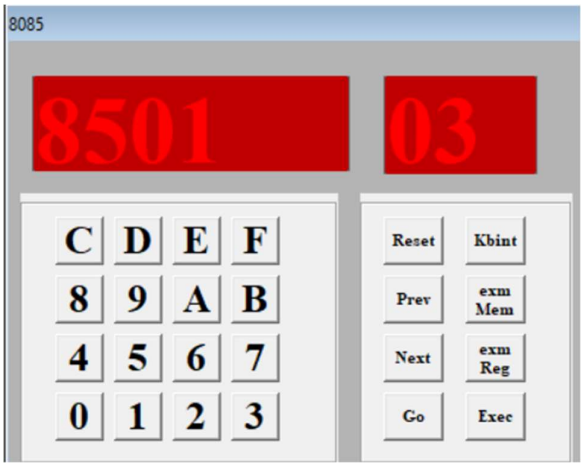


Figure 12.9 [8501] - 03

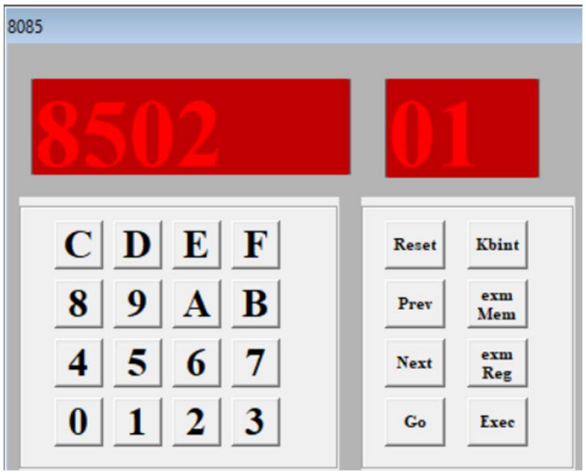


Figure 12.10 [8502] - 01

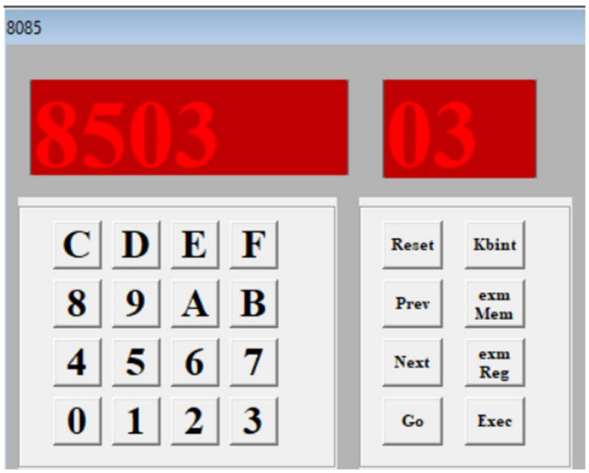


Figure 12.11 [8503] - 03

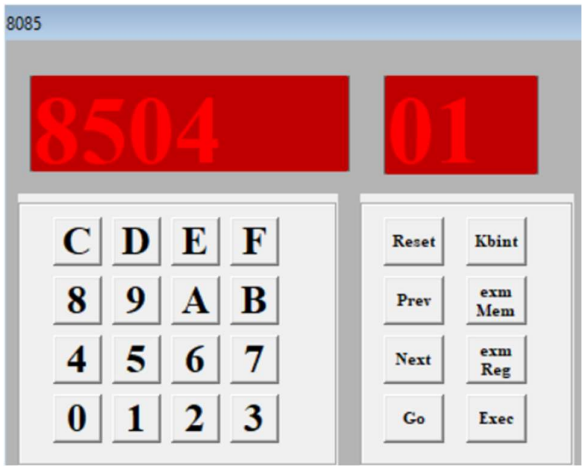


Figure 12.12 [8504] - 01

Figure 12.13 [8505] - 03

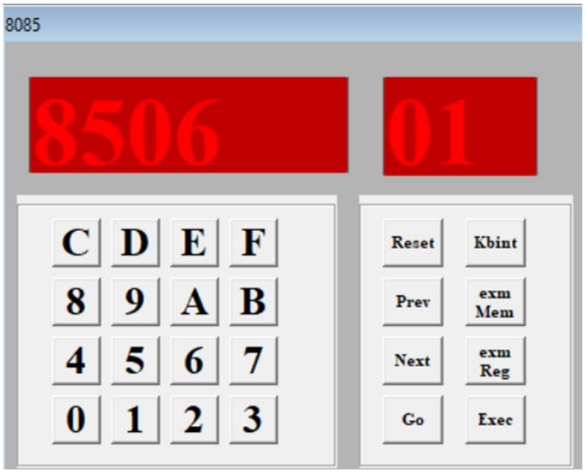


Figure 12.14 [8506] - 01

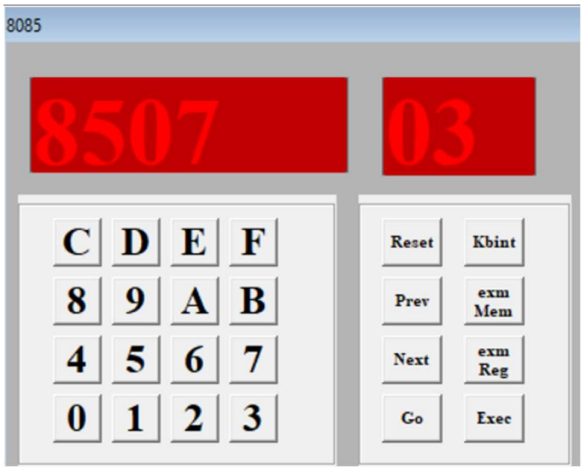


Figure 12.15 [8507] - 03

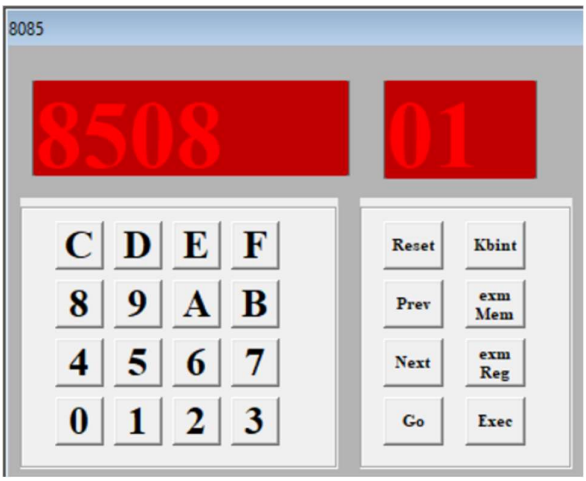


Figure 12.16 [8508] - 01

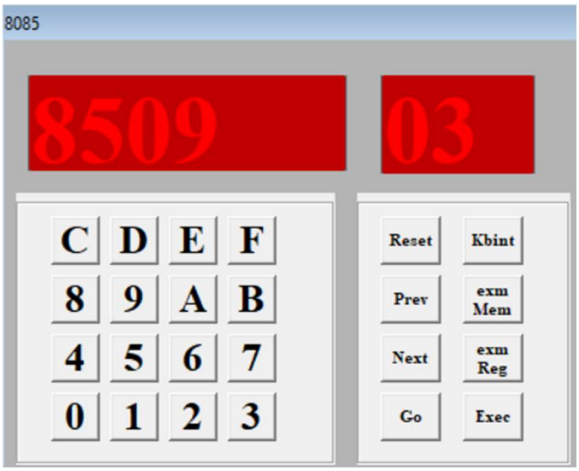


Figure 12.17 [8509] - 03

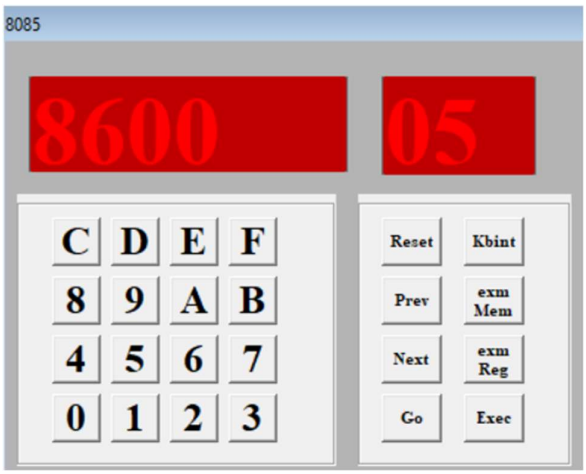


Figure 12.18 [8600] - 05

Input

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

[8506] – 01, [8507] – 03, [8508] – 01, [8509] – 03

Output

[8600] – 05

Program 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 13.1 Assembly language program to convert a BCD number into its equivalent binary

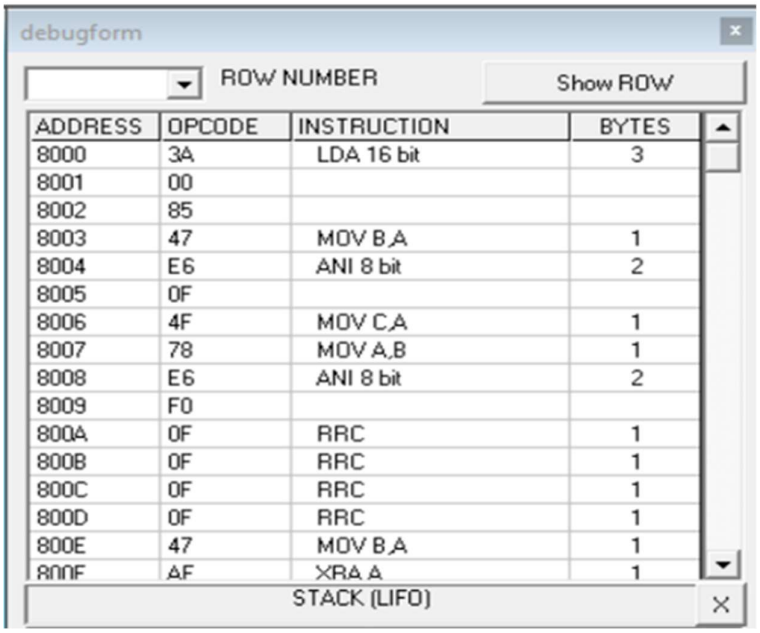


Figure 11.1 convert a BCD number into its equivalent binary

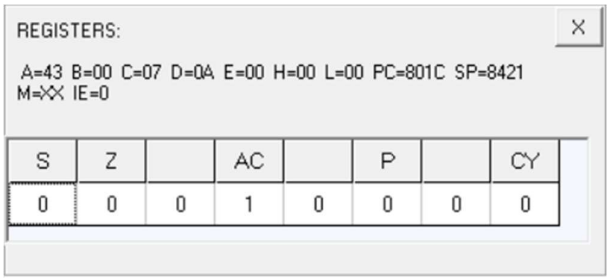


Figure 13.2 Values of Registers

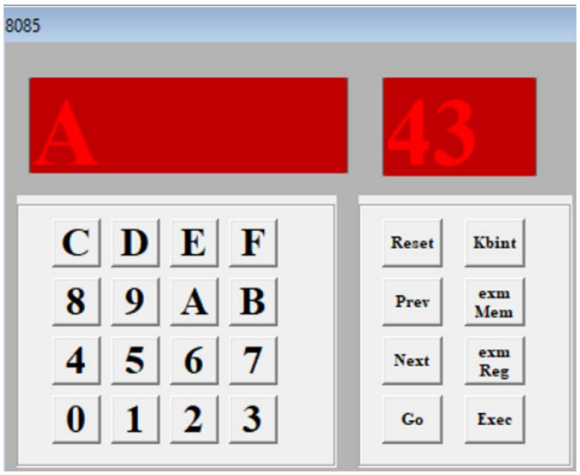


Figure 13.3 [A] - 43

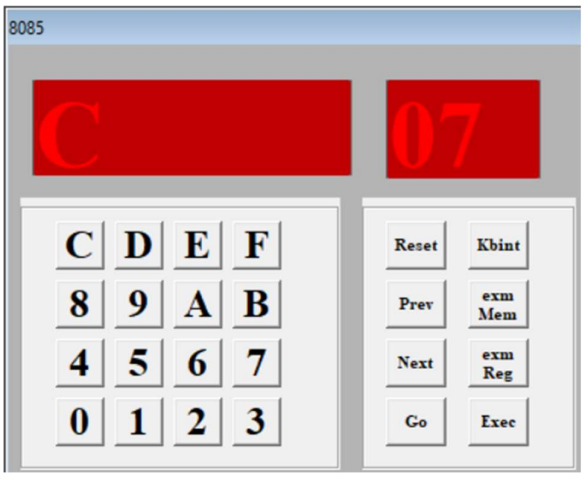


Figure 13.4 [C] - 07

Figure 13.5 [D] – 0A

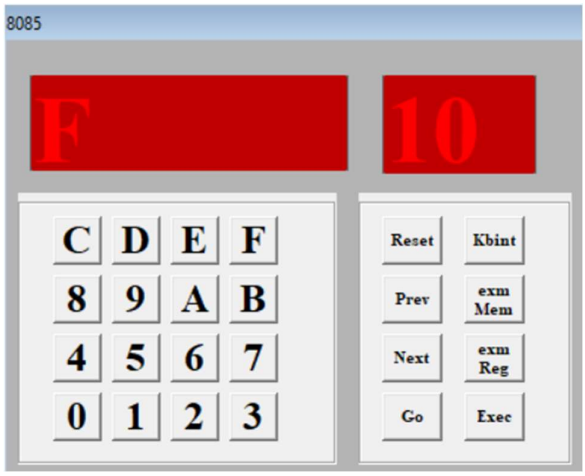


Figure 13.6 [F] - 10

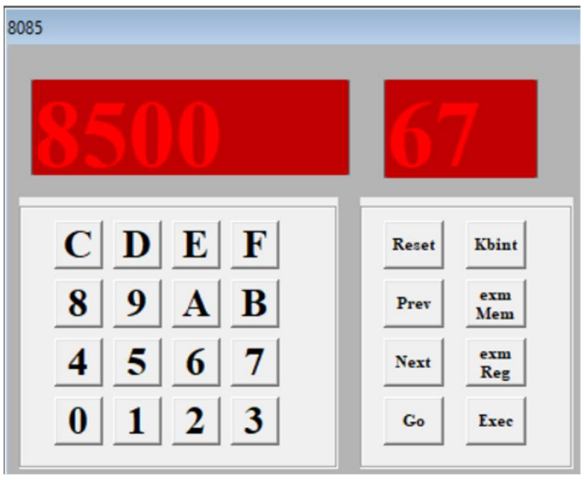


Figure 13.7 [8500] - 67

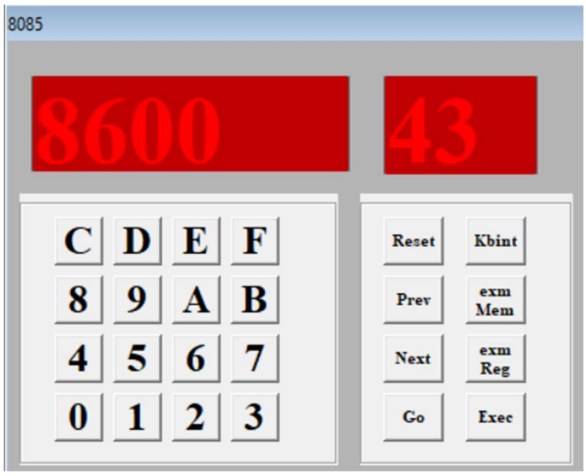


Figure 13.8 [8600] -43

Input

[8500] – 67

Input

[8600] – 43

Program No. 14

Aim: Write an ALP for exchange the contents of memory location.

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

Table 14.1 ALP for exchange the contents of memory location.

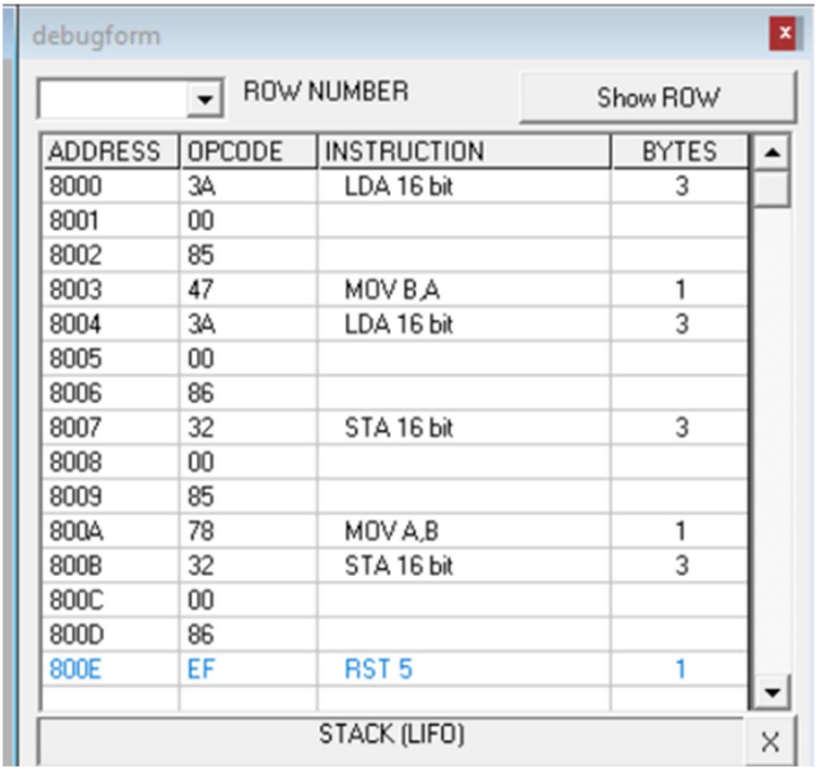


Figure 14.1 ALP for exchange the contents of memory location.

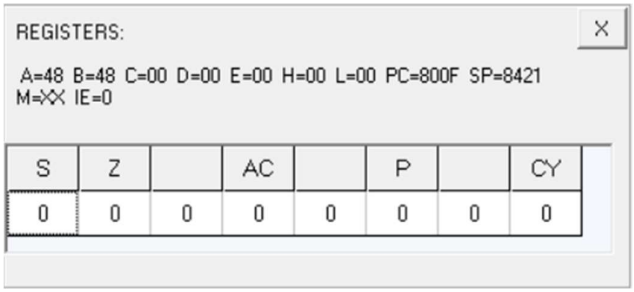


Figure 14.2 Values of Registers

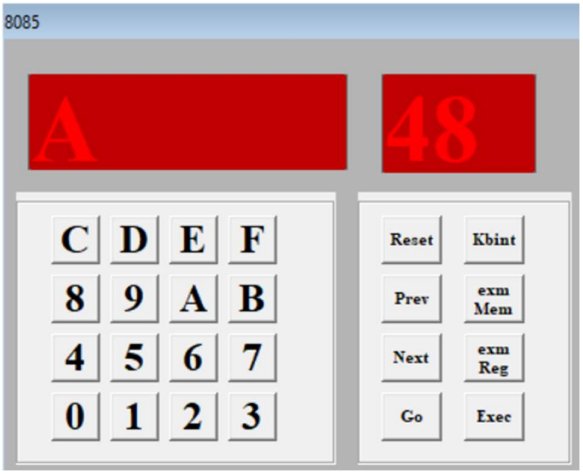


Figure 14.3 [A] - 48

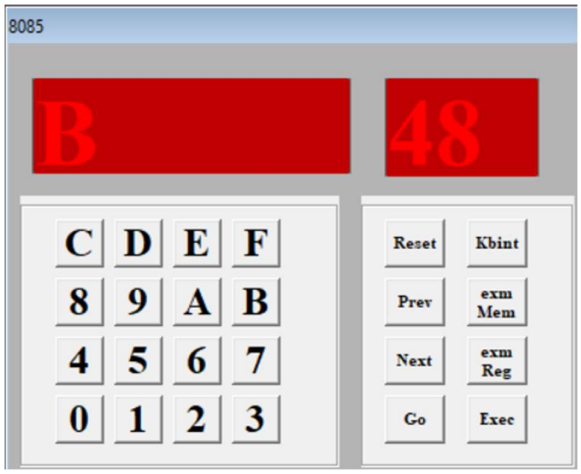


Figure 14.4 [B] - 48

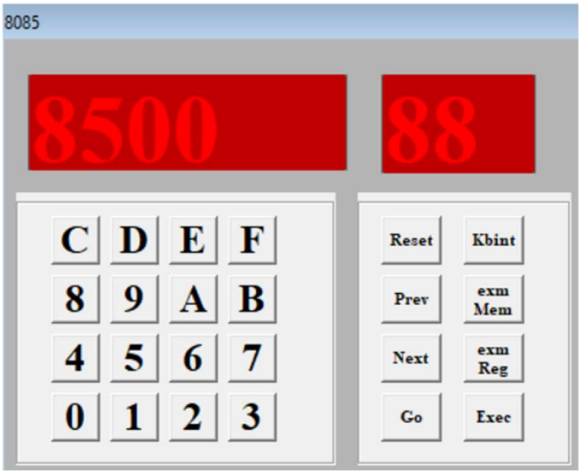


Figure 14.5 [8500] - 88

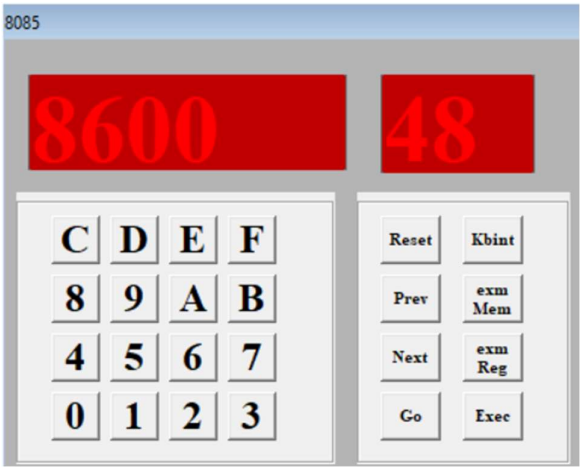


Figure 14.6 [8600] - 48

Input - [8500] – 48, [8600] – 88

Output - [8500] – 88, [8600] – 48

Program 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 15.1 program to find the largest number in an array of 10 elements.

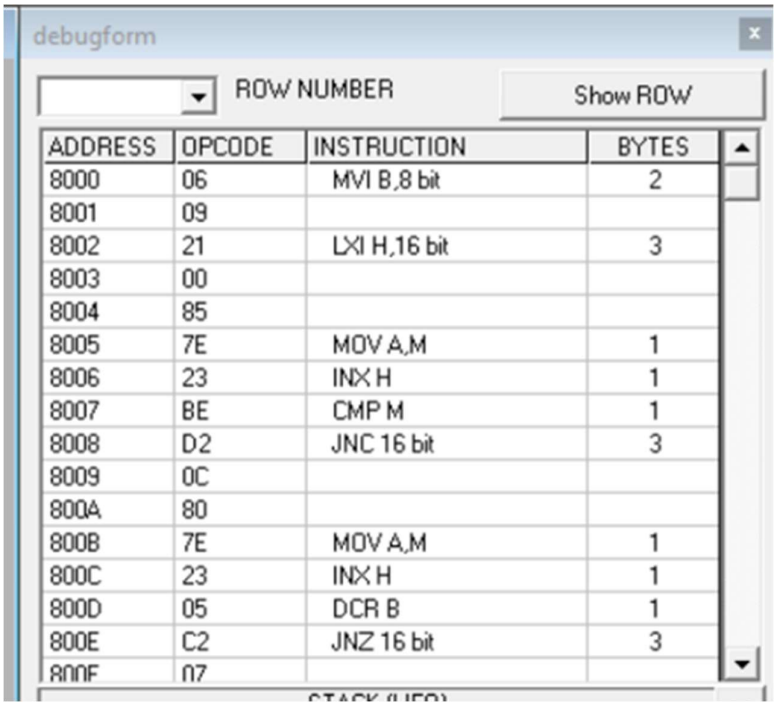


Figure 15.1 find the largest number in an array of 10 elements.

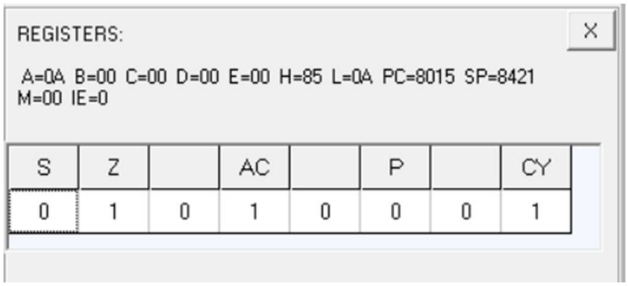


Figure 15.2 Values of Registers

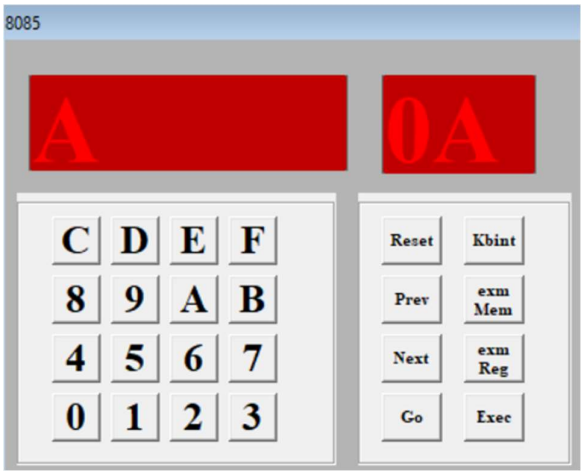


Figure 15.3 [A] – 0A

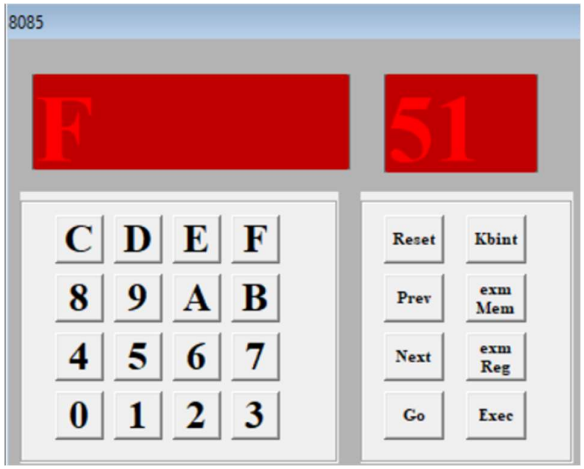


Figure 15.4 [F] - 51

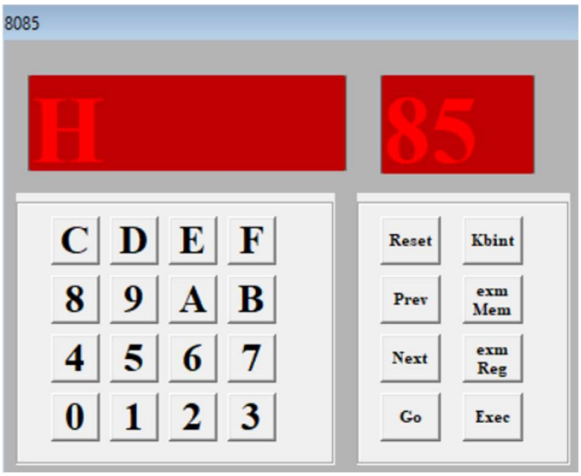


Figure 15.5 [H] - 85

Figure 15.6 [L] – 0A

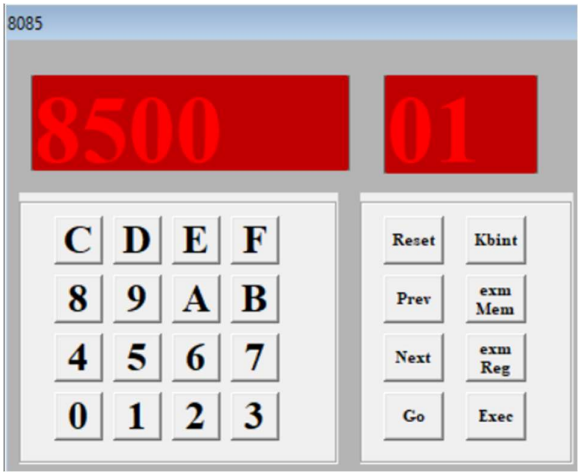


Figure 15.7 [8500] – 01

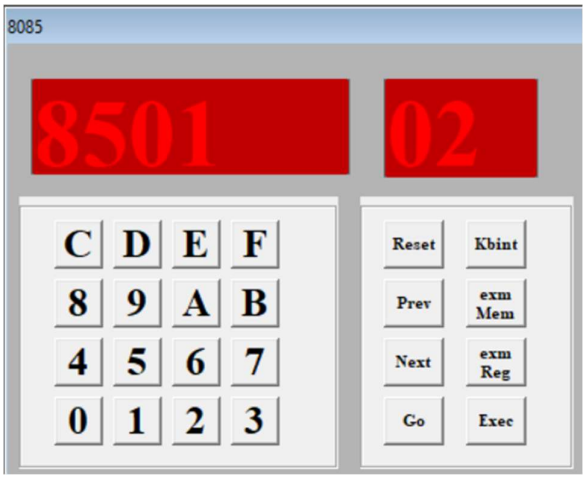


Figure 15.8 [8501] - 02

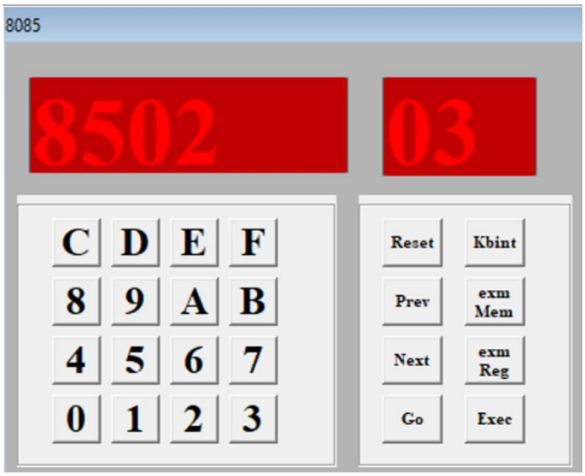


Figure 15.9 [8502] -03

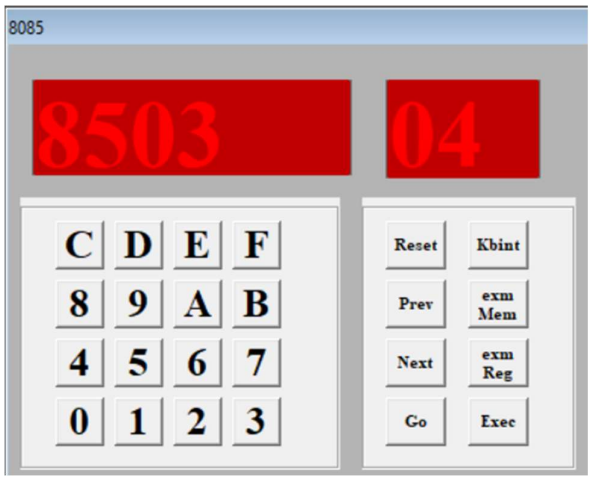


Figure 15.10 [8503] - 04

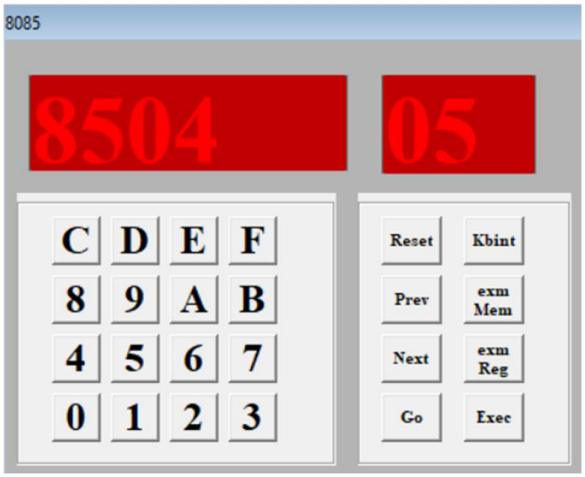


Figure 15.11 [8504] - 05

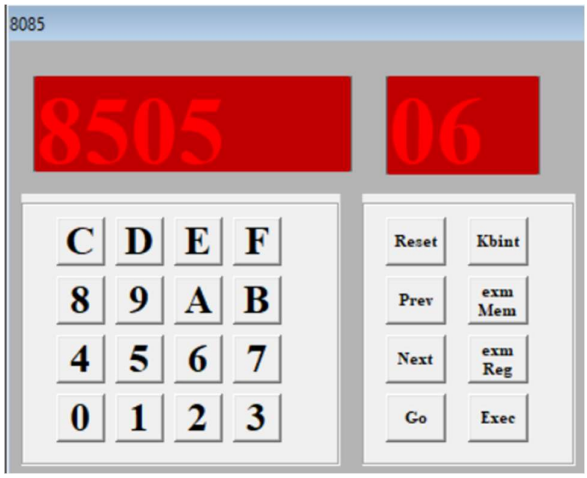


Figure 15.12 [8505] - 06

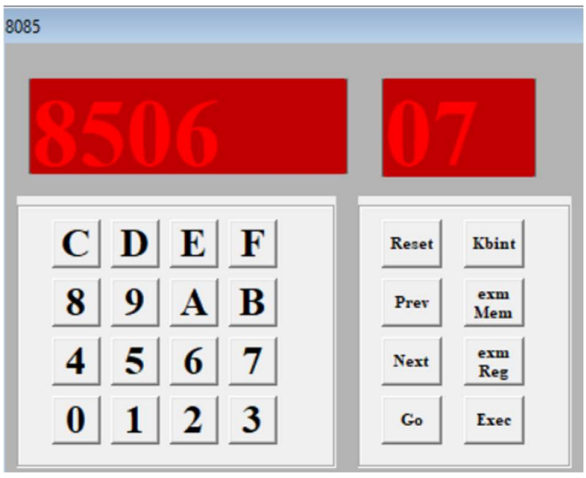


Figure 15.13 [8506] - 07

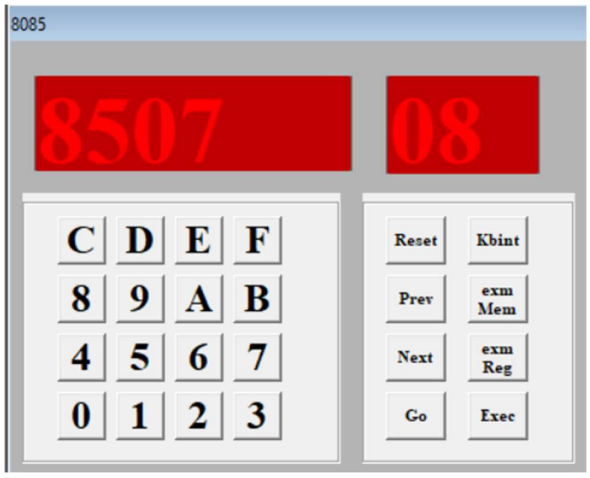


Figure 15.14 [8507] - 08

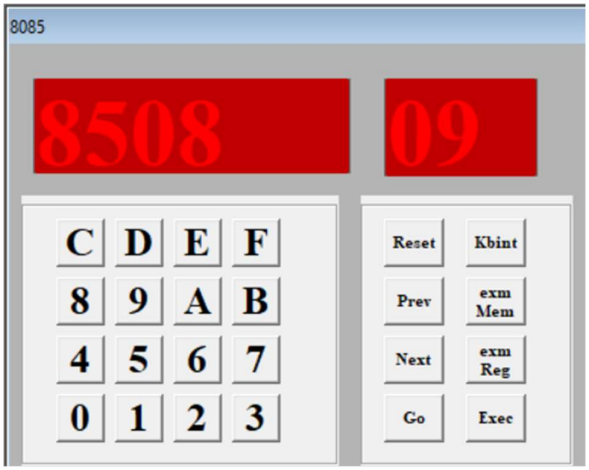


Figure 15.15 [8508] - 09

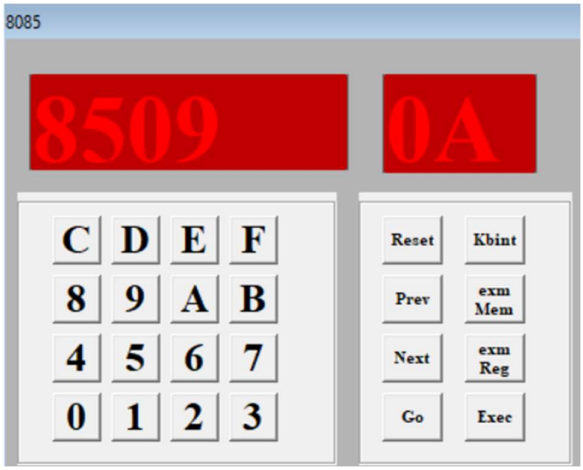


Figure 15.16 [8509] - 0A

Figure 15.17 [850A] – 0A

Input

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

Output

[850A] – 0A