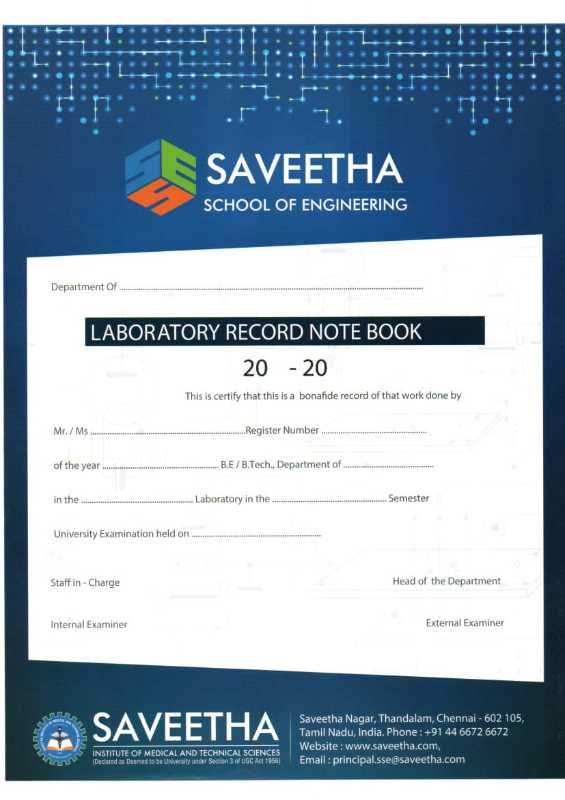
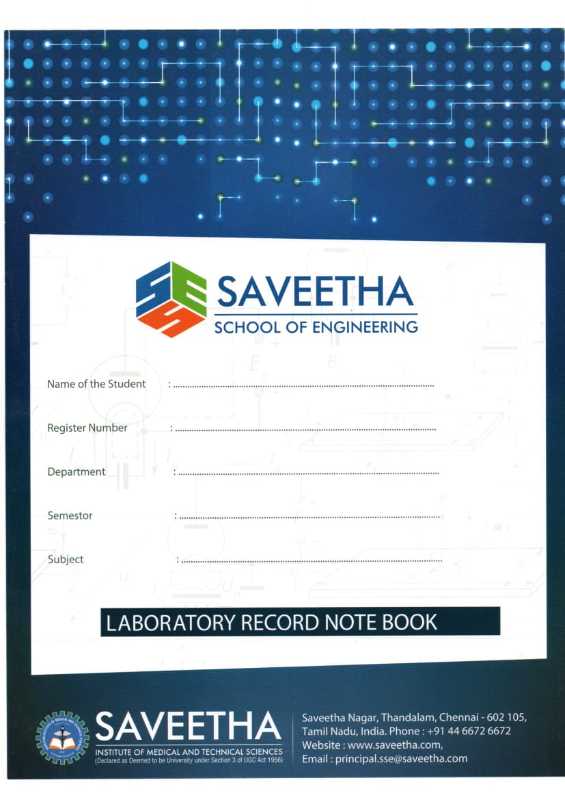
****

**Saveetha School of Engineering**

**Saveetha Institute of Medical and Technical Sciences**

**CSA12 - Computer Architecture**

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Exp.No.** | **TITLE** | **Page no.** | **Sign.** |
| **1.** | 8-BIT ADDITION | 5 |  |
| **2.** | 8-BIT SUBTRACTION | 6 |  |
| **3.** | 8-BIT MULTIPLICATION | 7 |  |
| **4.** | 8-BIT DIVISION | 8 |  |
| **5.** | 16 -BIT ADDITION | 9 |  |
| **6.** | 16-BIT SUBTRACTION | 10 |  |
| **7.** | 16-BIT MULTIPLICATION | 11 |  |
| **8.** | 16-BIT DIVISION | 12 |  |
| **9.** | FACTORIAL OF A GIVEN NUMBER | 13 |  |
| **10.** | LARGEST NUMBER IN AN ARRAY | 14 |  |
| **11.** | SMALLEST NUMBER IN AN ARRAY | 15 |  |
| **12.** | ASCENDING ORDER | 16 |  |
| **13.** | DESCENDING ORDER | 17 |  |
| **14.** | ADDITION OF N NUMBERS | 18 |  |
| **15.** | SWAPPING OF NUMBERS | 19 |  |
| **16.** | SQUARE OF NUMBER | 20 |  |
| **17.** | ONEs AND TWOs COMPLEMENT | 21 |  |
| **18.** | ROTATE LEFT OPERATION | 22 |  |
| **19** | ROTATE RIGHT OPERATION | 23 |  |
| **20** | LOGICAL OPERATIONS | 24 |  |
| **21** | DECIMAL TO BINARY CONVERSION | 25 |  |
| **22** | HEXA DECIMAL TO DECIMAL CONVERSION | 26 |  |
| **23** | DECIMAL TO OCTAL CONVERSION | 27 |  |
| **24** | BINARY TO DECIMAL CONVERSION | 28 |  |
| **25** | TWO STAGE PIPELINE | 29 |  |
| **26 PERFORMANCE** | CPU PERFORMANCE | 30 |  |
| **27** | TWO BIT HALF ADDER | 31 |  |
| **28** | TWO BIT HALF SUBTRACTOR | 32 |  |
| **29** | FULL ADDER | 33 |  |
| **30** | FULL SUBTRACTOR | 34 |  |

**8-BIT ADDITION**

**EXP NO: 1**

**AIM:**

To write an assembly language program to implement 8-bit addition using 8085 processor.

**ALGORITHM:**

1. Start the program by loading the first data into the accumulator.
2. Move the data to a register.
3. Get the second data and load it into the accumulator.
4. Add the two register contents.
5. Check for carry.
6. Store the value of sum and carry in the memory location.
7. Halt.

**PROGRAM:**

LDA 8500

MOV B, A

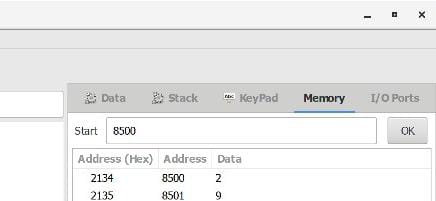
LDA 8501

ADD B

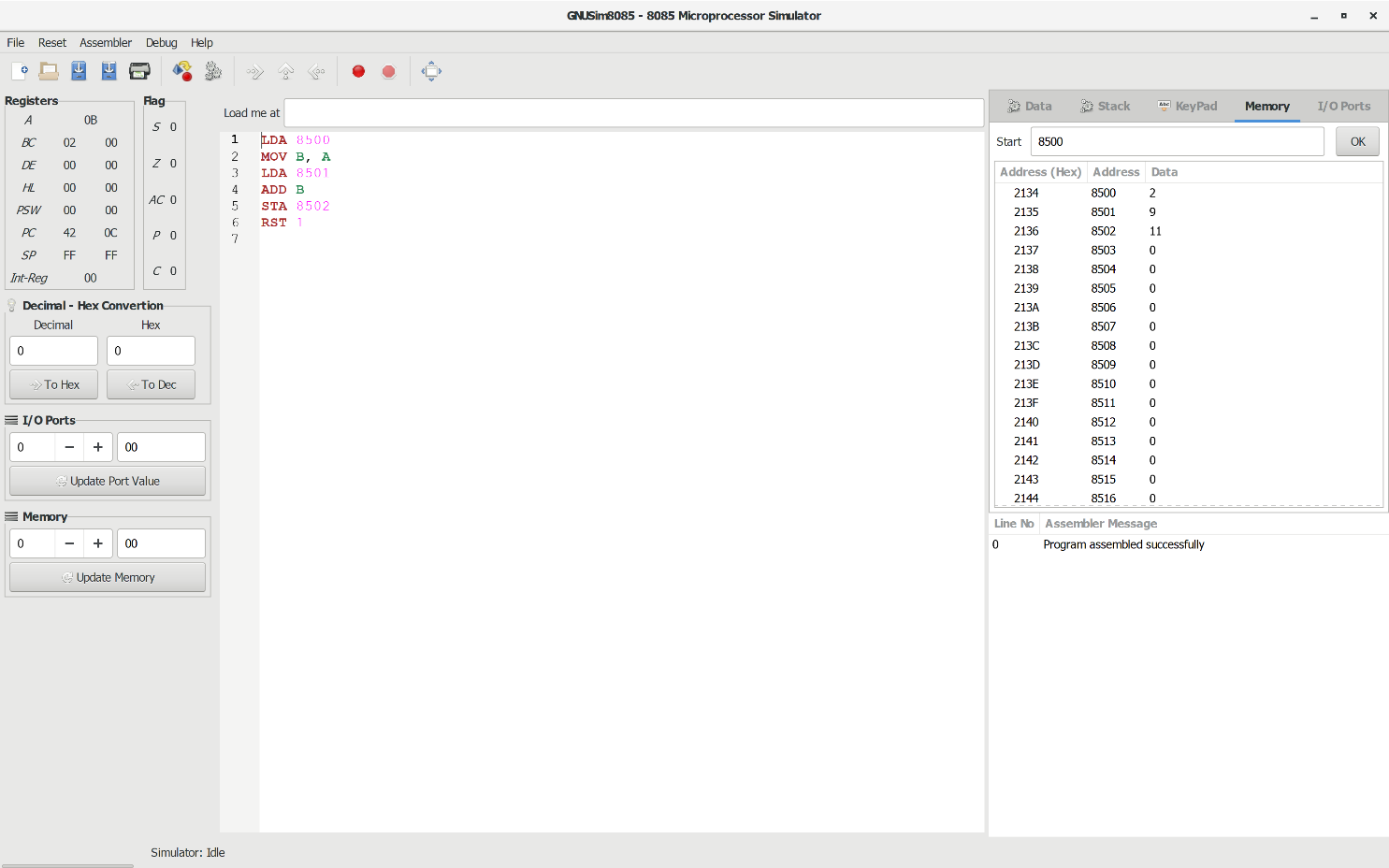
STA 8502

RST 1

**INPUT:**



**OUTPUT:**



**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**8-BIT SUBTRACTION**

**EXP NO: 2**

**AIM:** To write an assembly language program to implement 8-bit subtraction using 8085 processor.

**ALGORITHM:**

1. Start the program by loading the first data into the accumulator.
2. Move the data to a register.
3. Get the second data and load it into the accumulator.
4. Subtract the two register contents.
5. Check for borrow.
6. Store the difference and borrow in the memory location.
7. Halt.

**PROGRAM:**

LDA 8000

MOV B, A

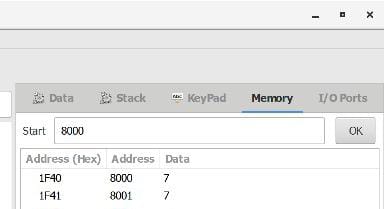
LDA 8001

SUB B

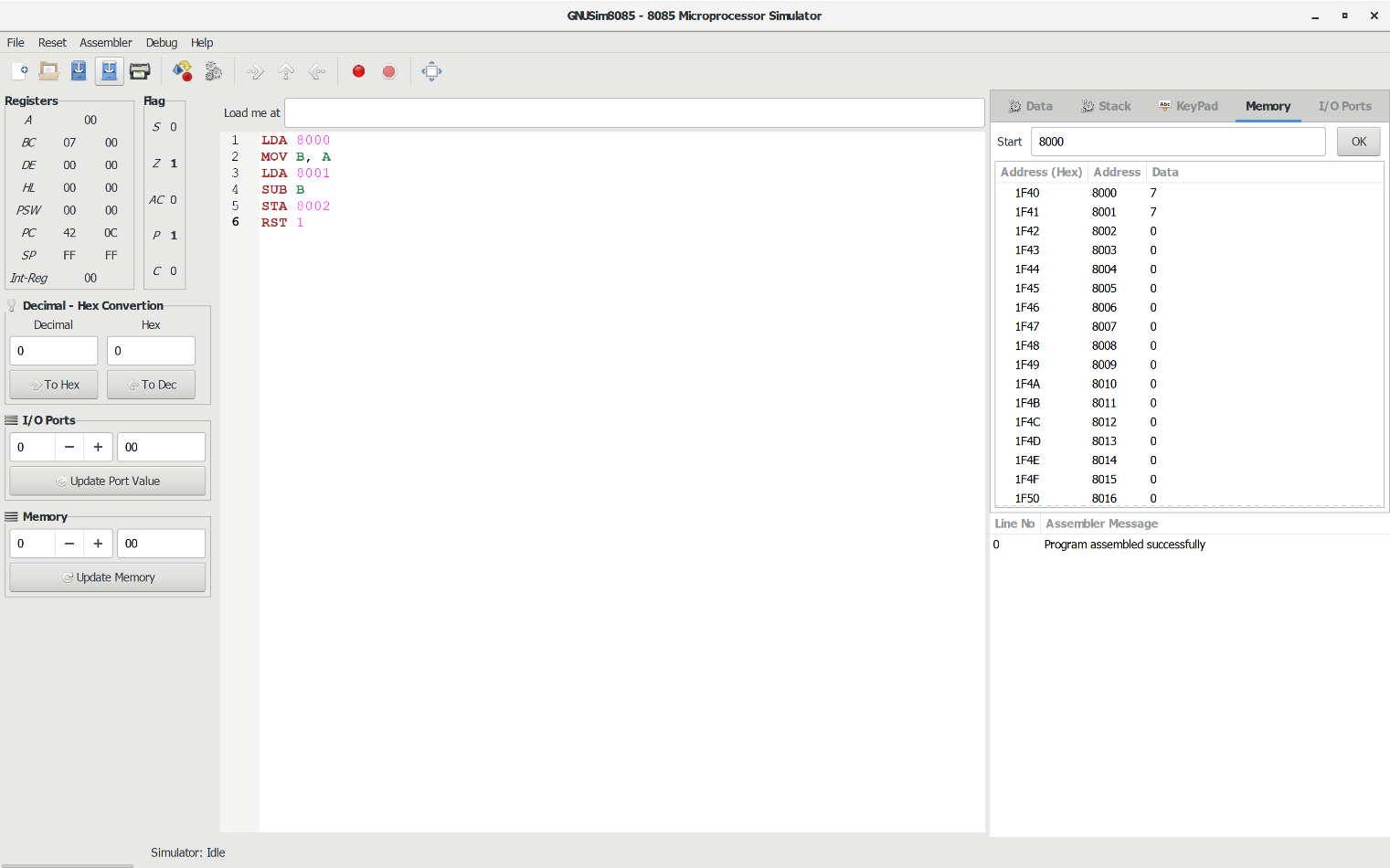
STA 8002

RST 1

**INPUT:**



**OUTPUT:**



**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**8-BIT MULTIPLICATION**

**EXP NO: 3**

**AIM:**

To write an assembly language program to implement 8-bit multiplication using 8085 processor.

**ALGORITHM:**

1. Start the program by loading a register pair with the address of memory location.
2. Move the data to a register.
3. Get the second data and load it into the accumulator.
4. Add the two register contents.
5. Increment the value of the carry.
6. Check whether the repeated addition is over.
7. Store the value of product and the carry in the memory location.
8. Halt.

**PROGRAM:**

LDA 8500

MOV B, A

LDA 8501

MOV C, A

CPI 00

JZ LOOP

XRA A

LOOP1: ADD B

DCR C

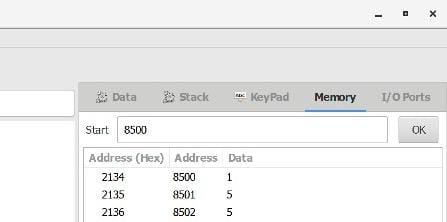
JZ LOOP

JMP LOOP1

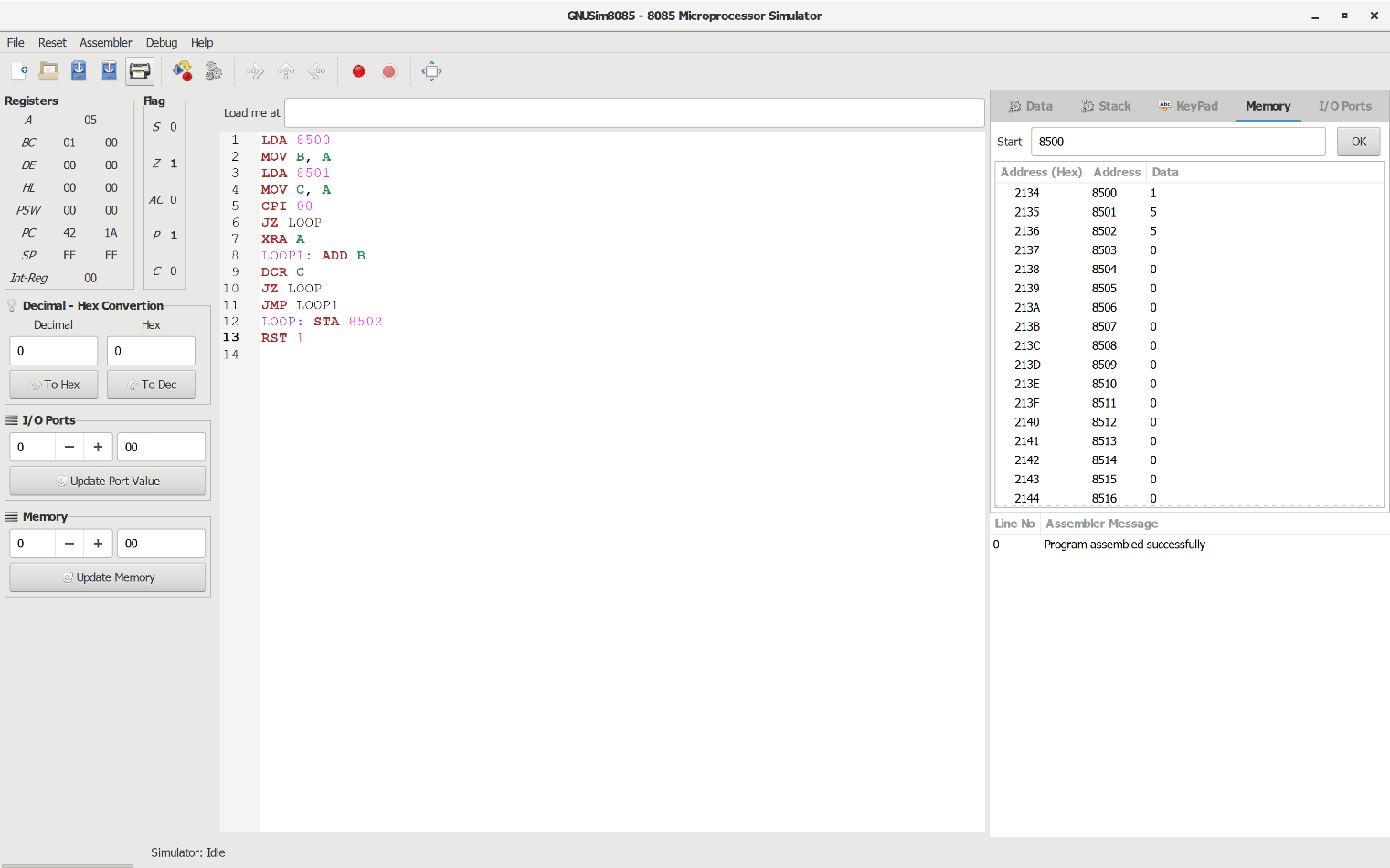
LOOP: STA 8502

RST 1

**INPUT:**

****

**OUTPUT:**



**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**8-BIT DIVISION**

**EXP NO: 4**

**AIM:**

To write an assembly language program to implement 8-bit division using 8085 processor.

**ALGORITHM:**

1. Start the program by loading a register pair with the address of memory location.
2. Move the data to a register.
3. Get the second data and load it into the accumulator.
4. Subtract the two register contents.
5. Increment the value of the carry.
6. Check whether the repeated subtraction is over.
7. Store the value of quotient and the reminder in the memory location.
8. Halt.

**PROGRAM:**

LDA 8501

MOV B, A

LDA 8500

MVI C,00

LOOP:CMP B

JC LOOP1

SUB B

INR C

JMP LOOP

STA 8503

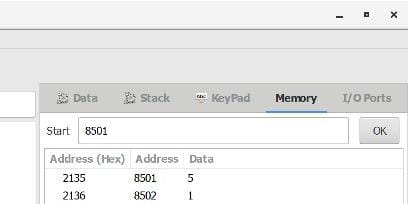
DCR C

MOV A, C

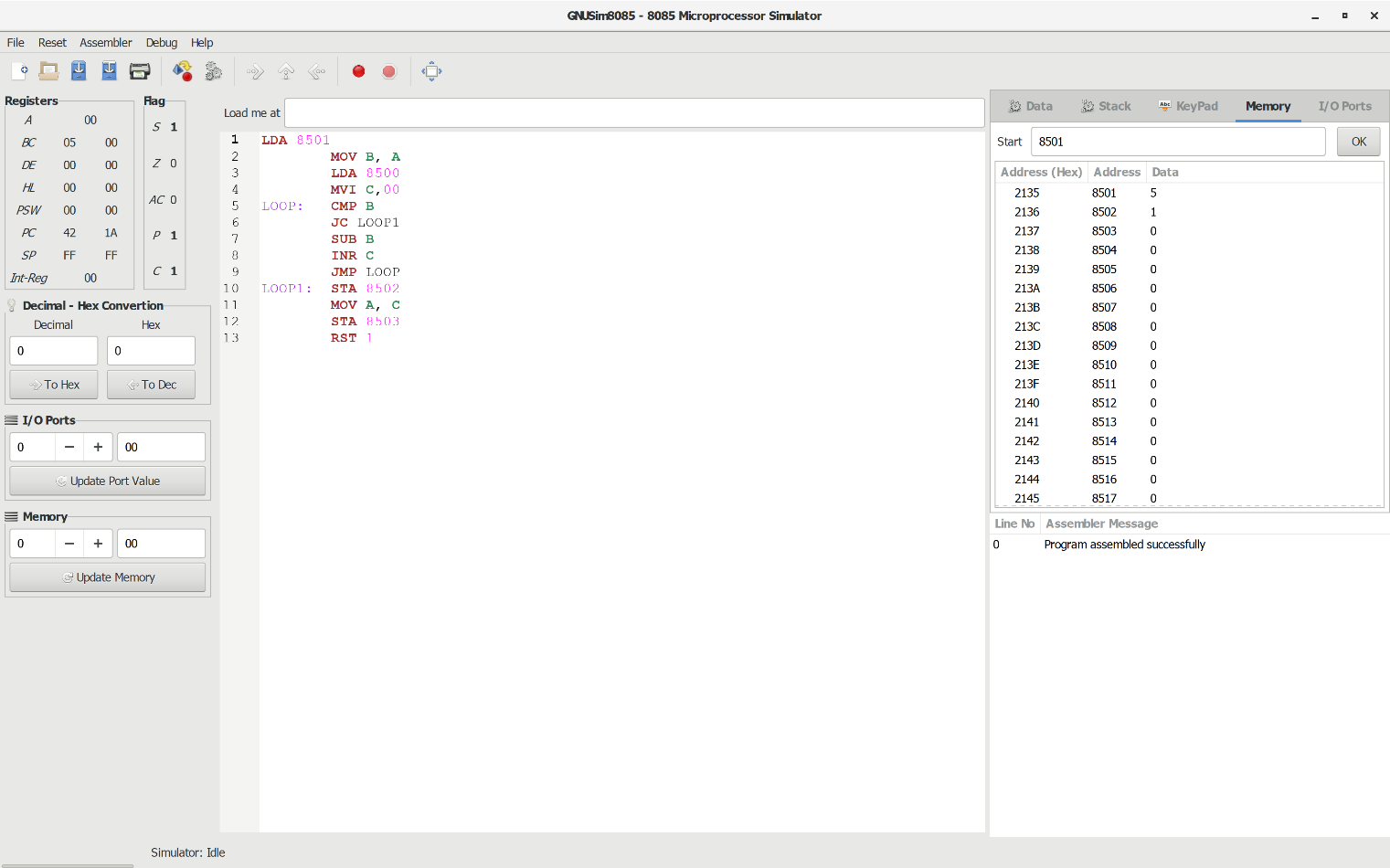
LOOP1: STA 8502

RST 1

**INPUT:**

****

**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**16-BIT ADDITION**

**EXP NO: 5**

**AIM:**

To write an assembly language program to implement 16-bit addition using 8085 processor.

**ALGORITHM:**

1) Start the program by loading a register pair with address of 1st number.

2)     Copy the data to another register pair.

3)      Load the second number to the first register pair.

4)      Add the two register pair contents.

5)      Store the result in memory locations.

6)      Terminate the program.

**PROGRAM:**

LHLD 2500

XCHG

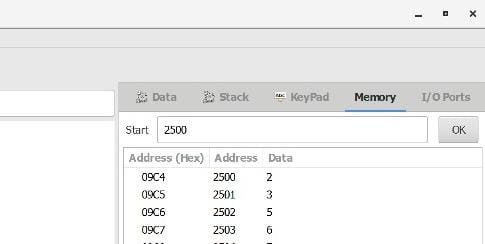
LHLD 2502

DAD D

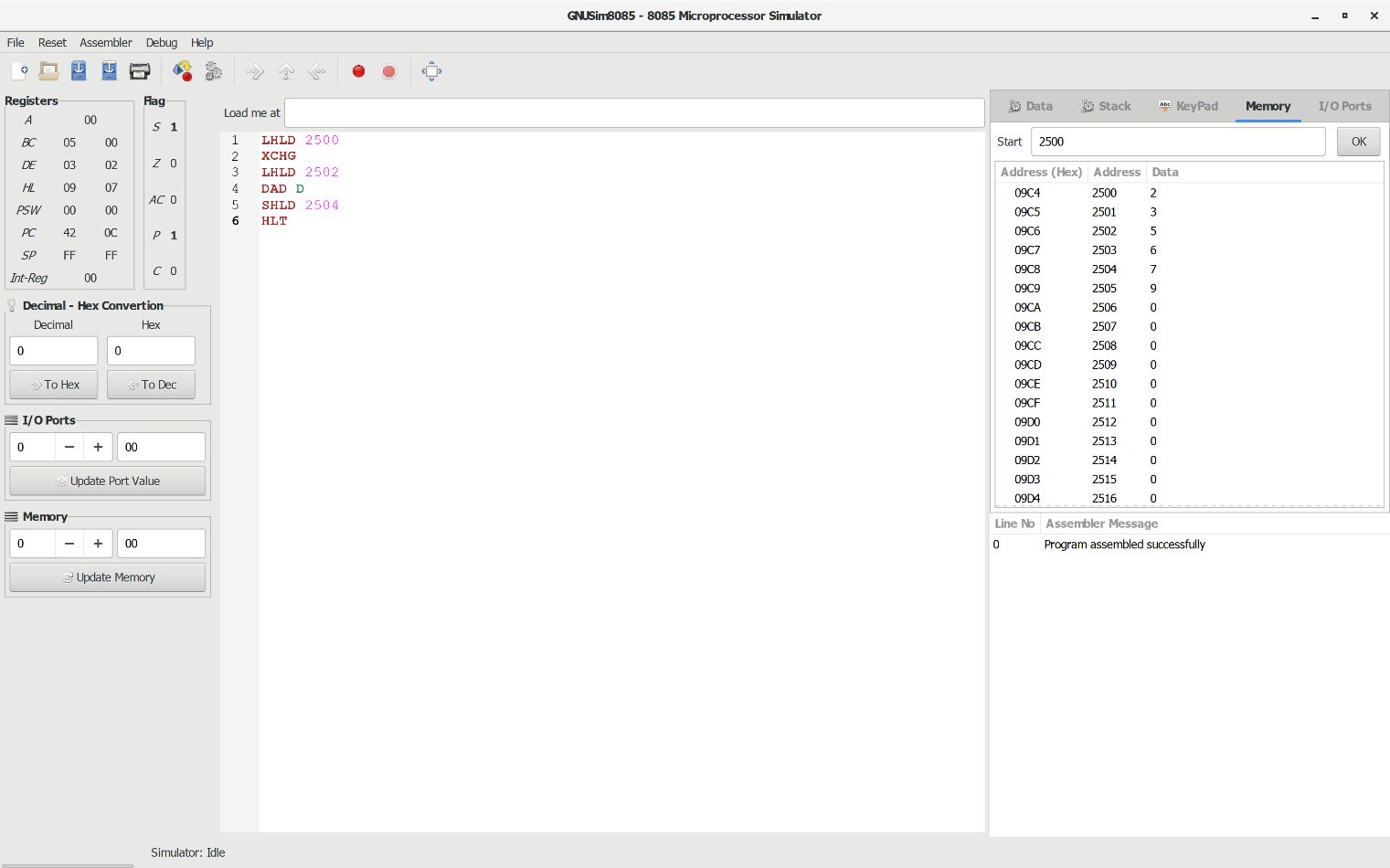
SHLD 2504

HLT

**INPUT:**

****

**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**16-BIT SUBTRACTION**

**EXP NO: 6**

**AIM:**

To write an assembly language program to implement 16-bit subtraction using 8085 processor.

**ALGORITHM:**

1) Start the program by loading a register pair with address of 1st number.

2)     Copy the data to another register pair.

3)      Load the second number to first registre pair.

4)     Subtract the two register pair contents.

5)      Check for borrow.

6)      Store the value of difference and borrow in memory locations.

7)      End.

**PROGRAM:**

LHLD 2050

XCHG

LHLD 2052

MVI C,00

MOV A, E

SUB L

STA 2054

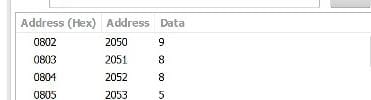
MOV A, D

SUB H

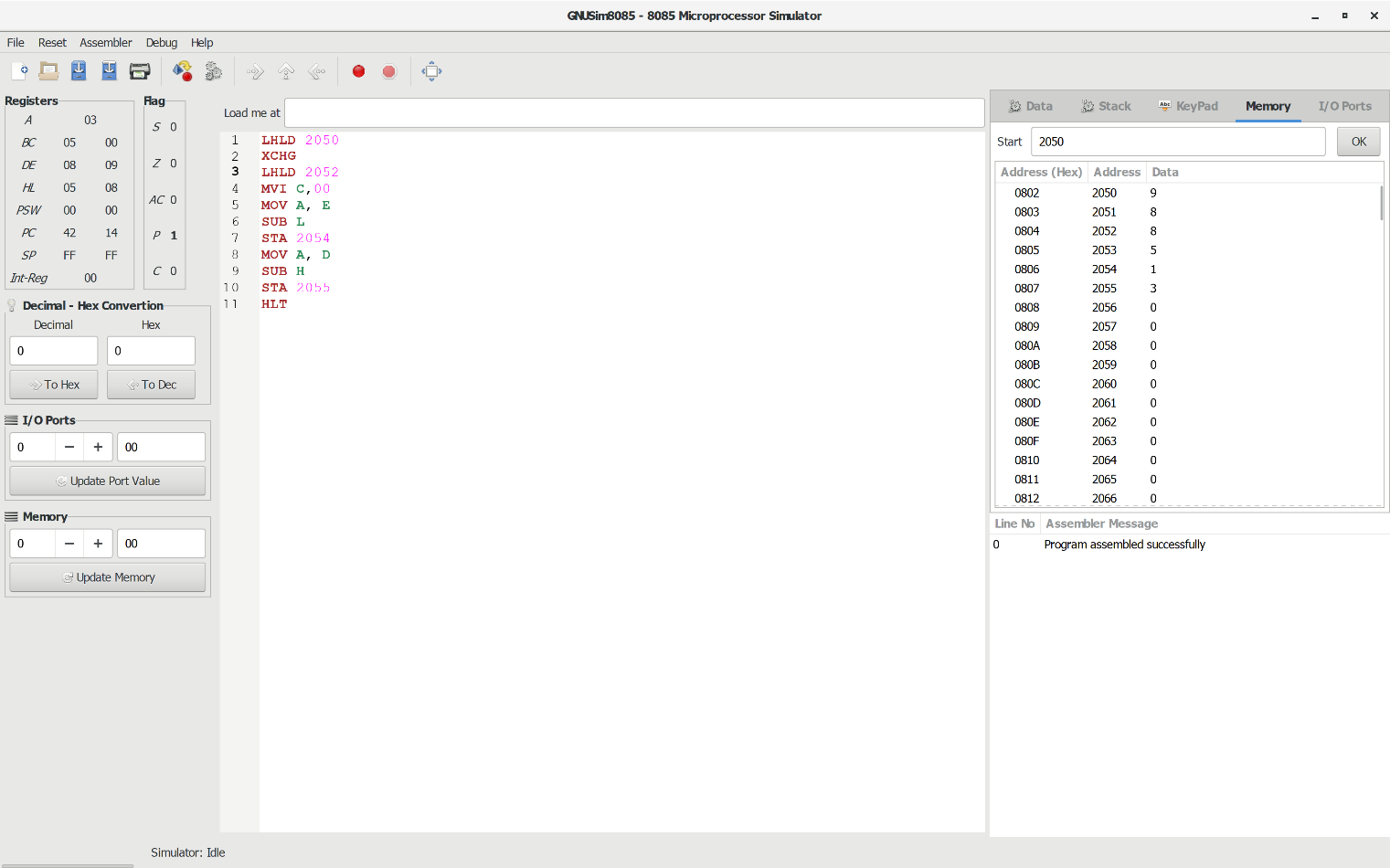
STA 2055

HLT

**INPUT:**

****

**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**16-BIT MULTIPLICATION**

**EXP NO: 7**

**AIM:**

To write an assembly language program to implement 16-bit multiplication using 8085 processor.

**ALGORITHM:**

1. Load the first data in HL pair.
2. Move content of HL pair to stack pointer.
3. Load the second data in HL pair and move it to DE.
4. Make H register as 00H and L register as 00H.
5. ADD HL pair and stack pointer.
6. Check for carry if carry increment it by 1 else move to next step.
7. Then move E to A and perform OR operation with accumulator and register D.
8. The value of operation is zero, then store the value else go to step 3.

**PROGRAM:**

LHLD­­­­ 2050

SPHL

LHLD 2052

XCHG

LXI H,0000H

LXI B,0000H

AGAIN: DAD SP

JNC START

INX B

START: DCX D

MOV A,E

ORA D

JNZ AGAIN

SHLD 2054

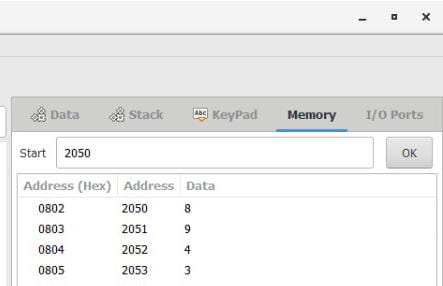
MOV L,C

MOV H,B

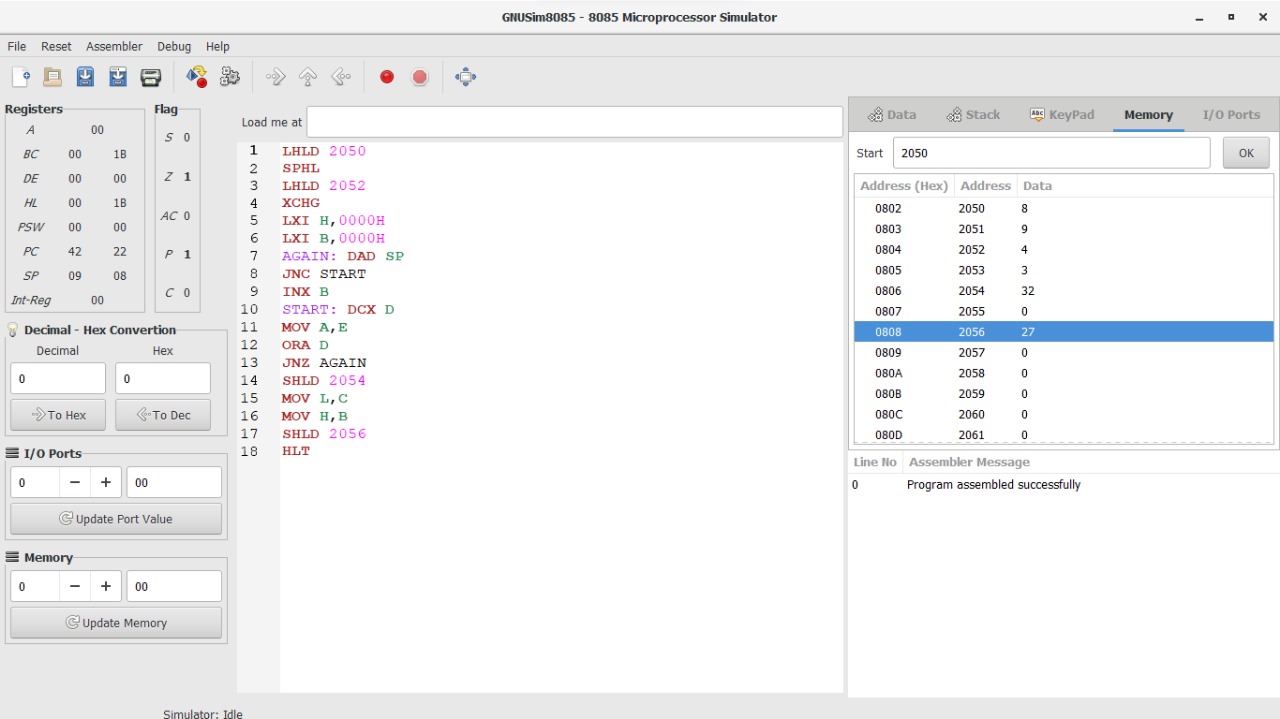
SHLD 2056

HLT

**INPUT:**

****

**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**16-BIT DIVISION**

**EXP NO: 8**

**AIM:**

To write an assembly language program to implement 16-bit divided by 8-bit using 8085 processor.

**ALGORITHM:**

1. Read dividend (16 bit)
2. Read divisor
3. count <- 8
4. Left shift dividend
5. Subtract divisor from upper 8-bits of dividend
6. If CS = 1 go to 9
7. Restore dividend
8. Increment lower 8-bits of dividend
9. count <- count - 1
10. If count = 0 go to 5
11. Store upper 8-bit dividend as remainder and lower 8-bit as quotient
12. Stop

**PROGRAM:**

LDA 8501

MOV B,A

LDA 8500

MVI C,00

LOOP:CMP B

JC LOOP1

SUB B

INR C

JMP LOOP

STA 8503

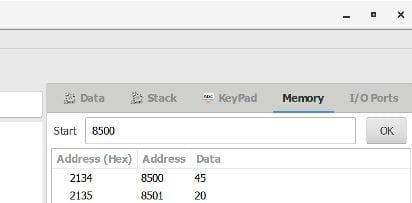
DCR C

MOV A,C

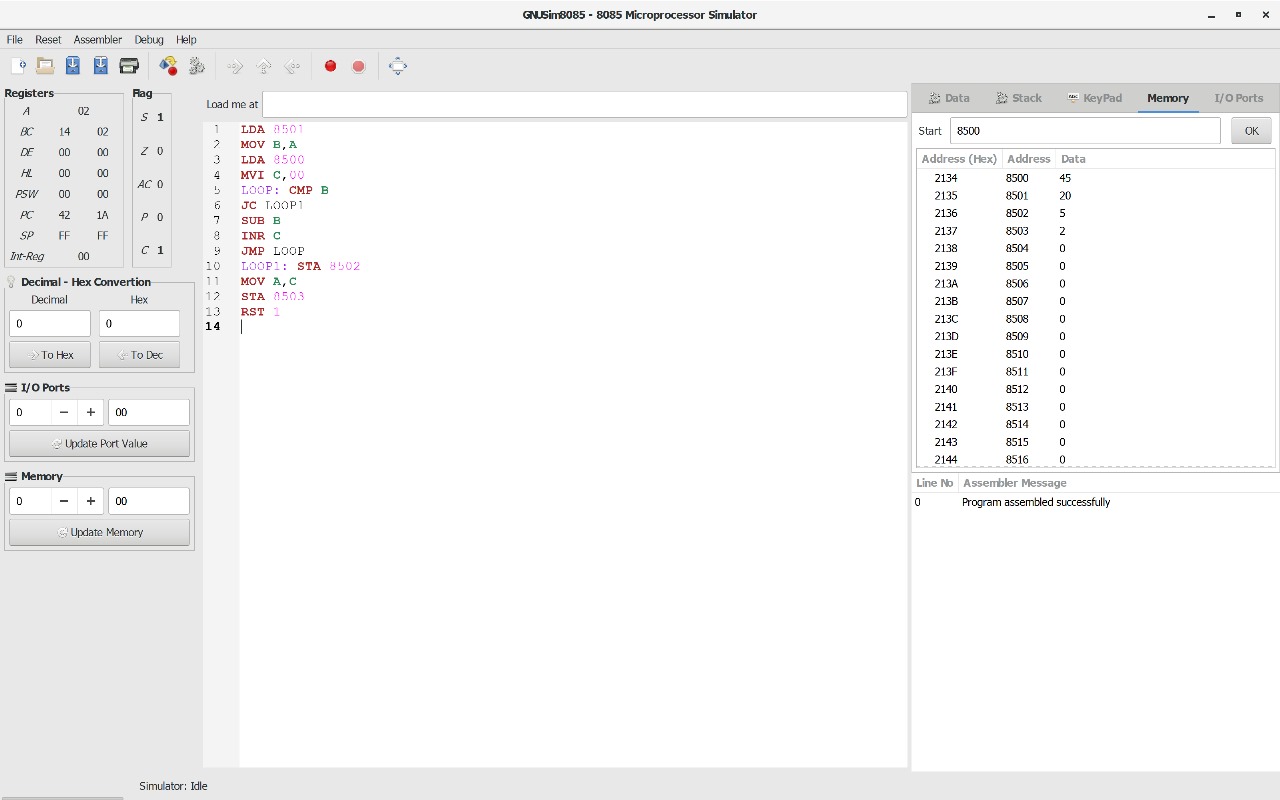
LOOP1: STA 8502

RST 1

**INPUT:**

****

**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**FACTORIAL OF A GIVEN NUMBER**

**EXP NO: 9**

**AIM:**

To find the factorial of a given number using 8085 microprocessor.

**ALGORITHM:**

1. Load the data into register B
2. To start multiplication set D to 01H
3. Jump to step 7
4. Decrements B to multiply previous number
5. Jump to step 3 till value of B>0
6. Take memory pointer to next location and store result
7. Load E with contents of B and clear accumulator
8. Repeatedly add contents of D to accumulator E times
9. Store accumulator content to D
10. Go to step 4

**PROGRAM:**

LDA 2001

MOV B,A

MVI C,#01

MVI E,#01

LOOP: MOV D,C

MVI A,00H

LP: ADD E

DCR D

JNZ LP

MOV E,A

INR C

DCR B

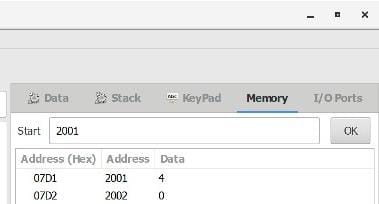
JNZ LOOP

MOV A,E

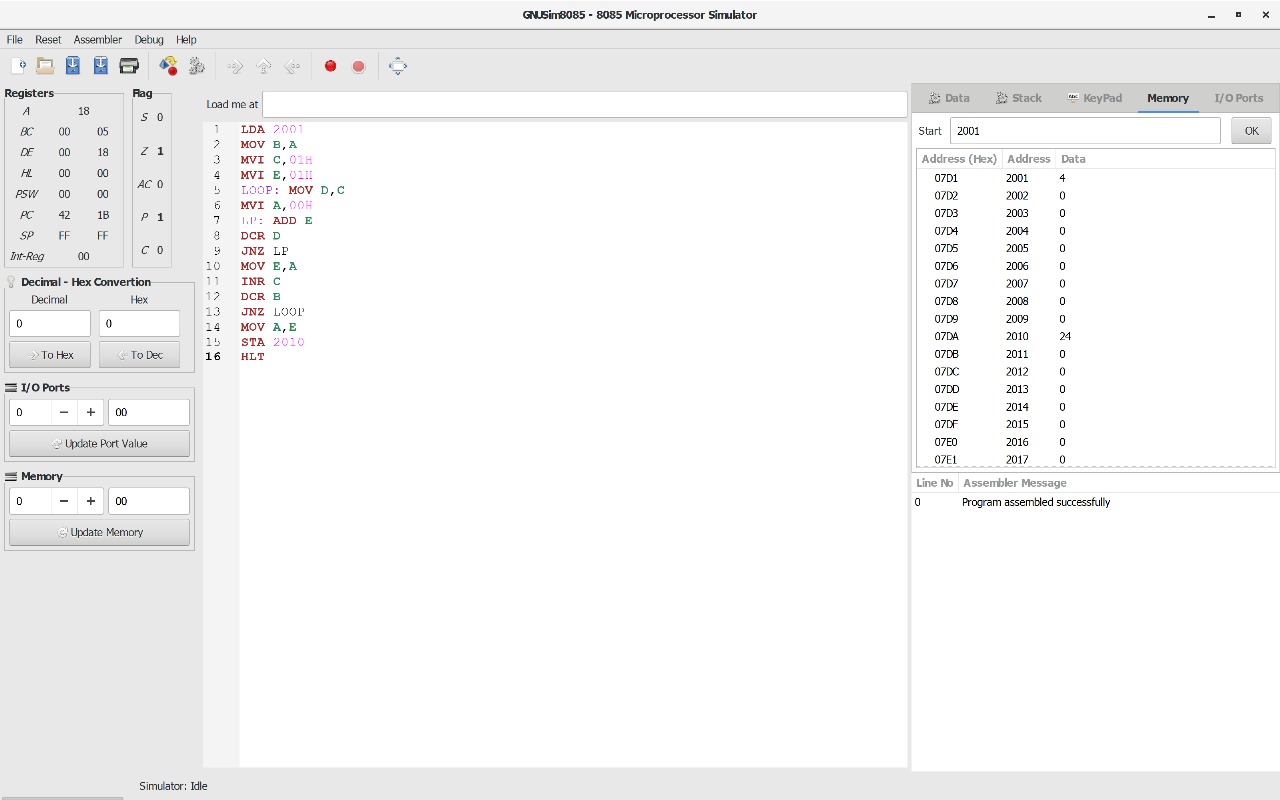
STA 2010

HLT

**INPUT:**



**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**LARGEST NUMBER IN AN ARRAY**

**EXP NO: 10**

**AIM:**

To find the largest number from an array using 8085 processor.

**ALGORITHM:**

1. Load the address of the first element of the array in HL pair.
2. Move the count to B register.
3. Increment the pointer.
4. Get the first data in A register.
5. Decrement the count.
6. Increment the pointer.
7. Compare the content of memory addressed by HL pair with that of A register.
8. If carry=0, go to step 10 or if carry=1 go to step 9
9. Move the content of memory addressed by HL to A register.
10. Decrement the count.

**PROGRAM:**

LXI H,2050

MOV C,M

DCR C

INX H

MOV A,M

LOOP1: INX H

CMP M

JNC LOOP

MOV A,M

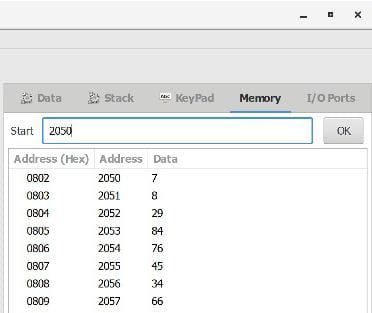
LOOP: DCR C

JNZ LOOP1

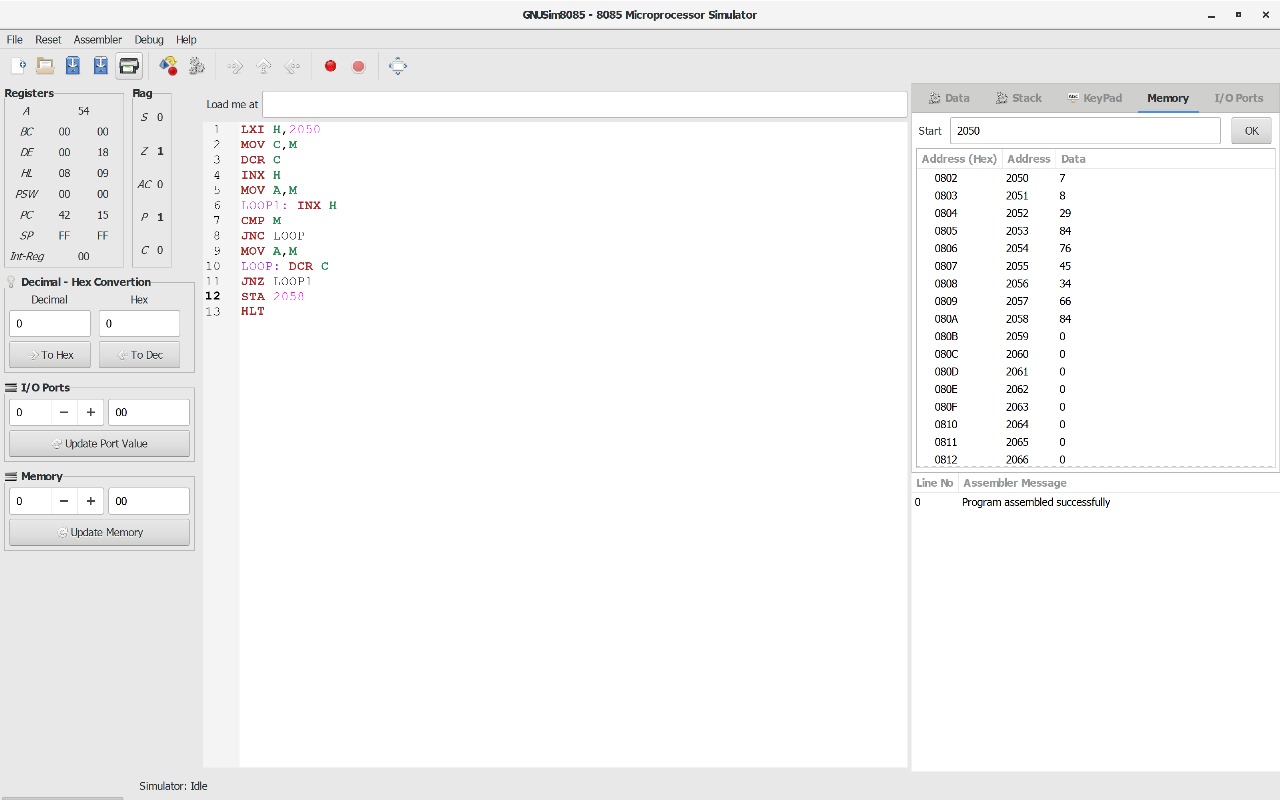
STA 2058

HLT

**INPUT:**



**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8086 processor simulator.

**SMALLEST NUMBER IN AN ARRAY**

**EXP NO: 11**

**AIM:**

To find the smallest number from an array using 8085 processor.

**ALGORITHM:**

1. Load the address of the first element of the array in HL pair.
2. Move the count to B register.
3. Increment the pointer.
4. Get the first data in A register.
5. Decrement the count.
6. Increment the pointer.
7. Compare the content of memory addressed by HL pair with that of A register.
8. If carry=1, go to step 10 or if carry=0 go to step 9
9. Move the content of memory addressed by HL to A register.
10. Decrement the count.

**PROGRAM:**

LXI H,2050

MOV C,M

DCR C

INX H

MOV A,M

LOOP1: INX H

CMP M

JC LOOP

MOV A,M

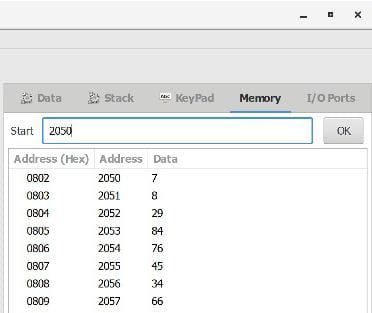
LOOP: DCR C

JNZ LOOP1

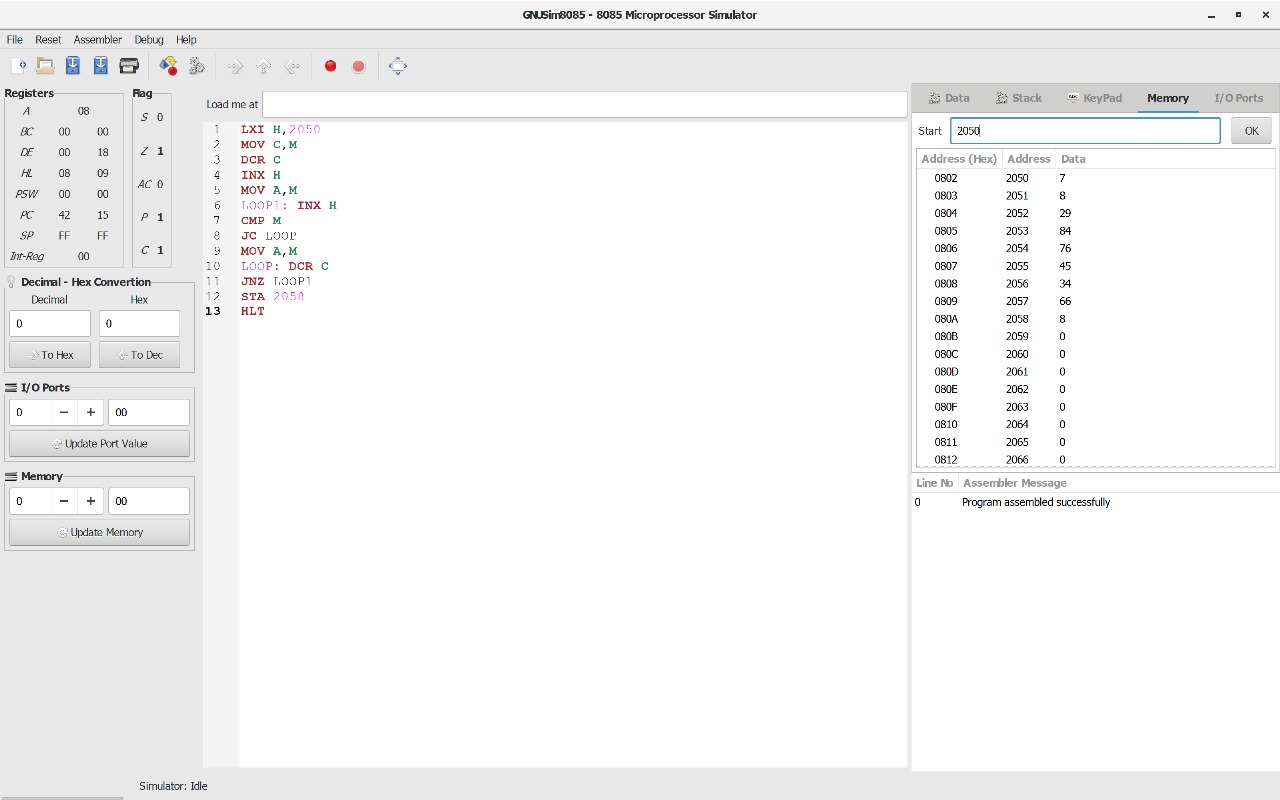
STA 2058

HLT

**INPUT:**



**OUTPUT:**



**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**ASCENDING ORDER**

**EXP NO: 12**

**AIM:**

To compute ascending order of an array using 8085 processor.

**ALGORITHM:**

1. Initialize HL pair as memory pointer.
2. Get the count at memory and load it into C register
3. Copy it in D register (for bubble sort (N-1)) times required).
4. Get the first value in A register.
5. Compare it with the value at next location.
6. If they are out of order, exchange the contents of A register and memory.
7. Decrement D register content by 1
8. Repeat step 5 and 7 till the value in D register become zero.
9. Decrement the C register content by 1.
10. Repeat steps 3 to 9 till the value in C register becomes zero.

**PROGRAM:**

LOOP: LXI H,3500

MVI D,00

MVI C,05

LOOP1: MOV A,M

INX H

CMP M

JC LOOP2

MOV B,M

MOV M,A

DCX H

MOV M,B

INX H

MVI D,01

LOOP2: DCR C

JNZ LOOP1

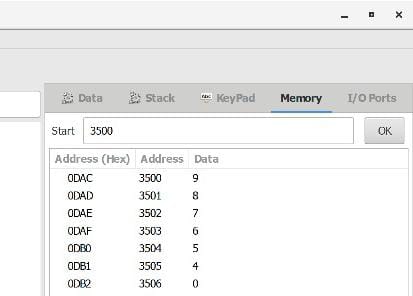
MOV A,D

RRC

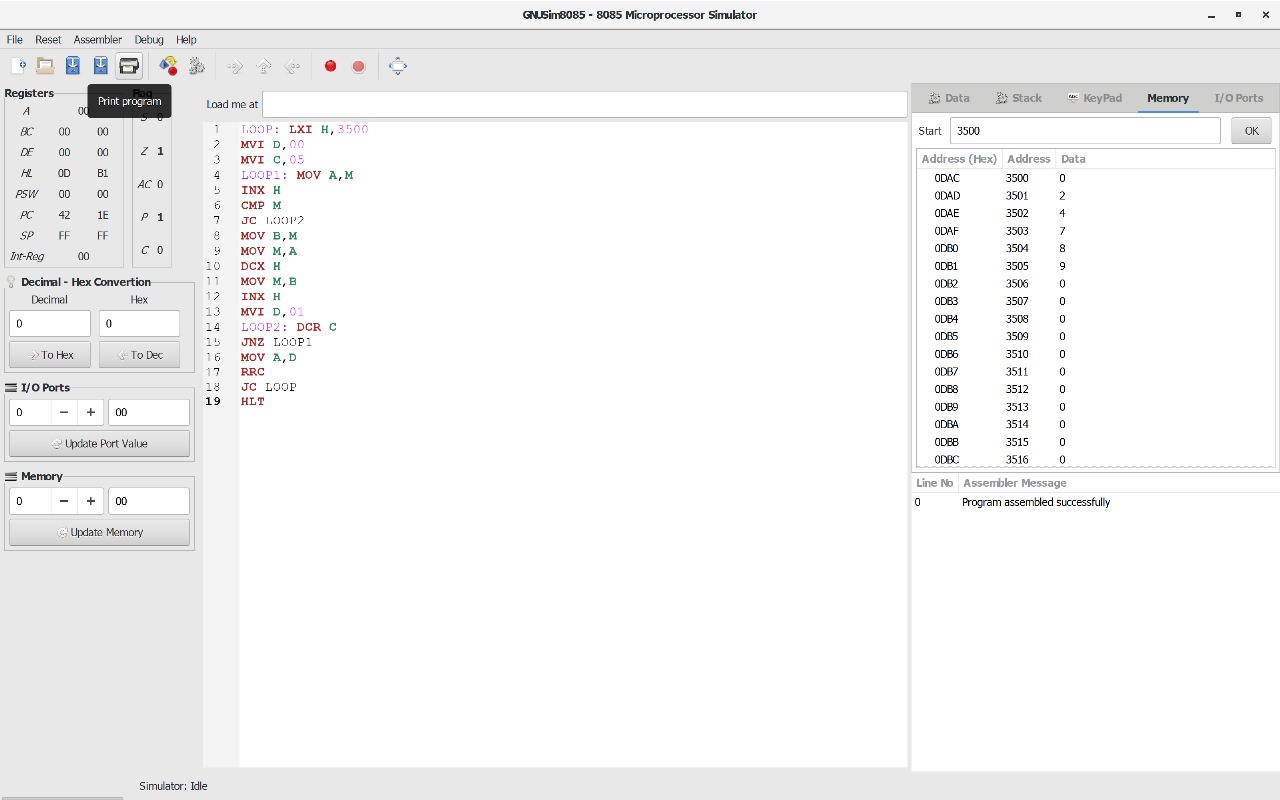
JC LOOP

HLT

**INPUT:**

****

**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**DESCENDING ORDER**

**EXP NO: 13**

**AIM:**

To compute descending order of an array using 8085 processor.

**ALGORITHM:**

1. Initialize HL pair as memory pointer.
2. Get the count at memory and load it into C register
3. Copy it in D register (for bubble sort (N-1)) times required).
4. Get the first value in A register.
5. Compare it with the value at next location.
6. If they are out of order, exchange the contents of A register and memory.
7. Decrement D register content by 1
8. Repeat step 5 and 7 till the value in D register become zero.
9. Decrement the C register content by 1.
10. Repeat steps 3 to 9 till the value in C register becomes zero.

**PROGRAM:**

LOOP: LXI H,3500

MVI D,00

MVI C,05

LOOP1: MOV A,M

INX H

CMP M

JNC LOOP2

MOV B,M

MOV M,A

DCX H

MOV M,B

INX H

MVI D,01

LOOP2: DCR C

JNZ LOOP1

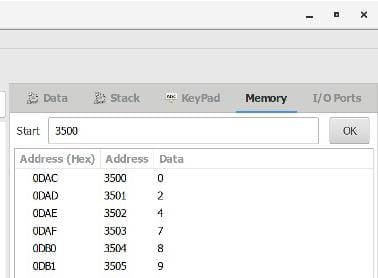
MOV A,D

RRC

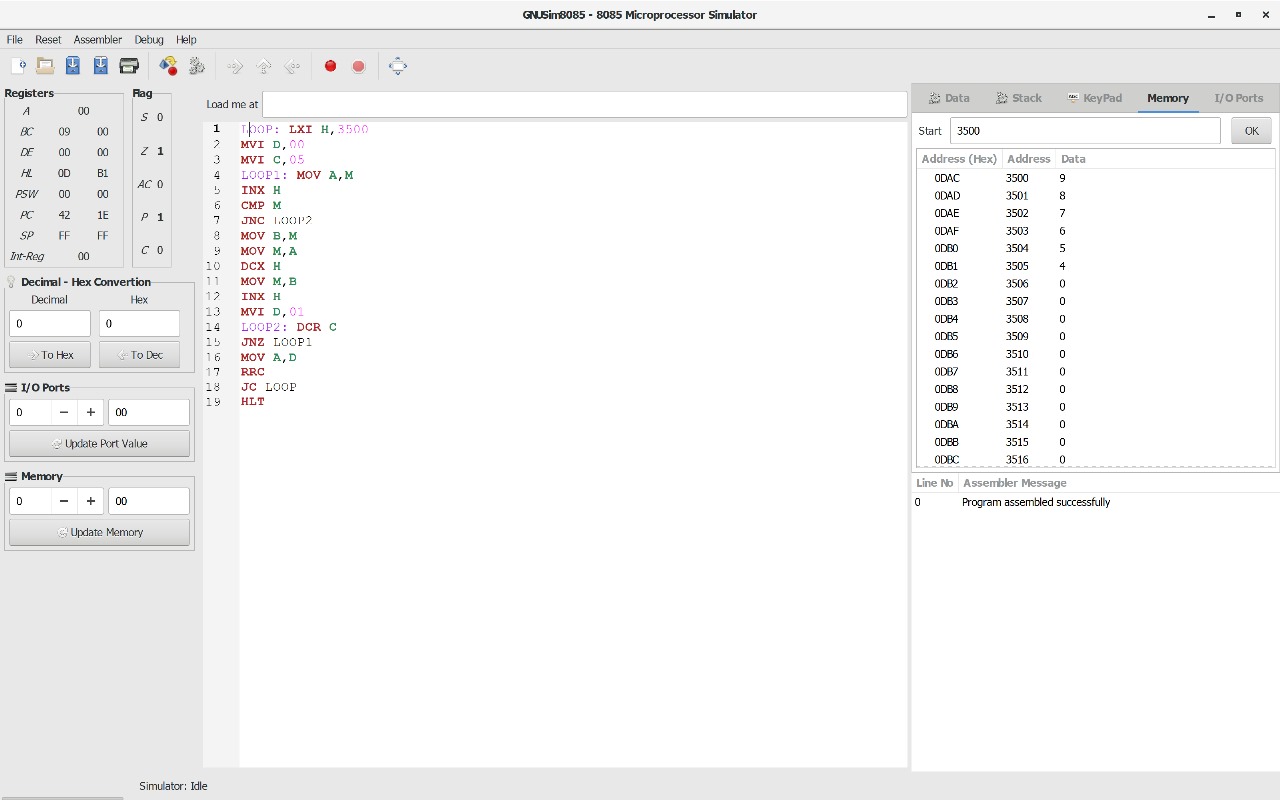
JC LOOP

HLT

**INPUT:**

****

**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**ADDITION OF N NUMBERS**

**EXP NO: 14**

**AIM:**

To compute addition of N numbers using 8085 processor.

**ALGORITHM:**

1. Load the base address of the array in HL register pair.
2. Load the memory with data to be added.
3. Take it as count.
4. Initialize the accumulator with 00.
5. Add content of accumulator with content of memory.
6. Decrement count.
7. Load count value to memory location.
8. Repeat step 5.
9. Check whether count has become 0.
10. Halt.

**PROGRAM:**

LXI H,8000

MOV C,M

MVI A,00

MOV B,A

LOOP: ADD C

JNC SKIP

INR B

SKIP: DCR C

JNZ LOOP

LXI H,8007

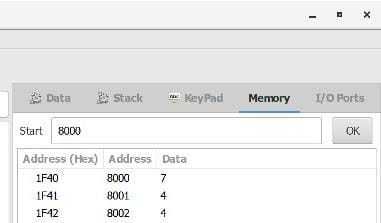
MOV M,A

INX H

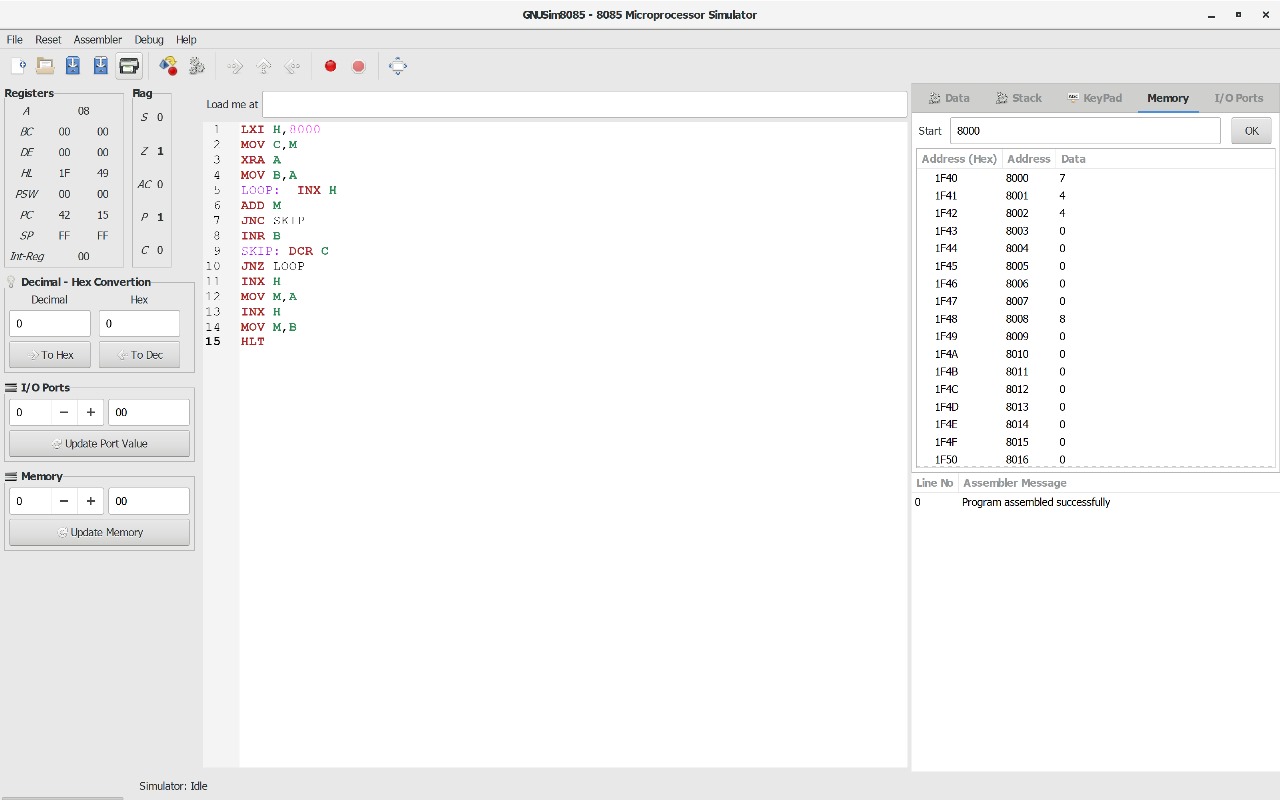
MOV M,B

HLT

**INPUT:**

****

**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**SWAPPING OF NUMBERS**

**EXP NO: 15**

**AIM:**

To compute swapping of numbers using 8085 processor.

**ALGORITHM:**

1. Load a 8-bit number from memory location into accumulator.
2. Move value of accumulator into register H.
3. Load a 8-bit number from next memory location into accumulator.
4. Move value of accumulator into register D.
5. Exchange both the registers pairs.
6. Halt

**PROGRAM:**

LDA 2001

MOV B,A

LDA 2002

MOV C,A

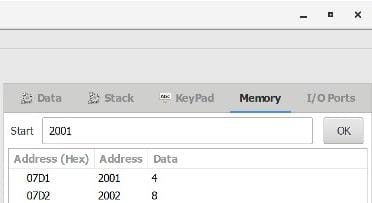
STA 2003

MOV A,B

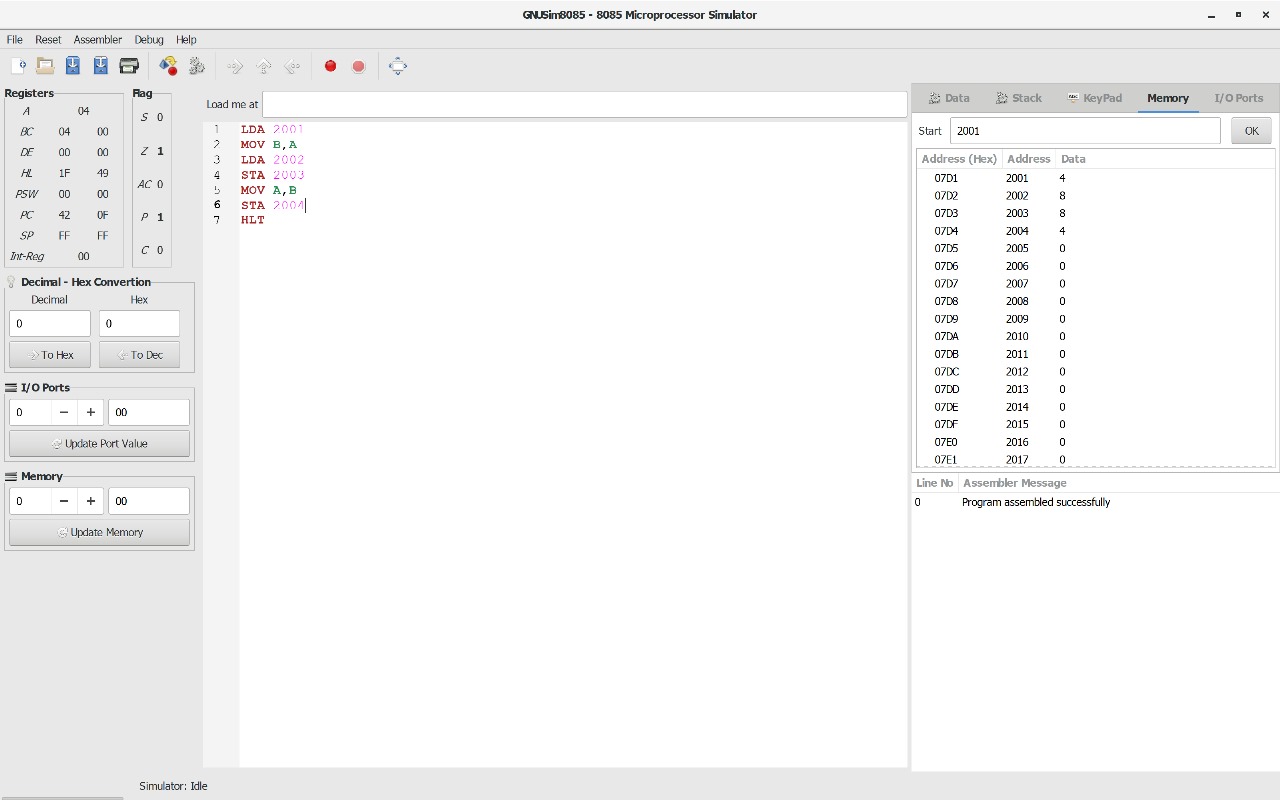
STA 2004

HLT

**INPUT:**

****

**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**SQUARE OF NUMBER**

**EXP NO: 16**

**AIM:**

To compute square of number using 8085 processor.

**ALGORITHM:**

1. Load the base address of the array in HL register pair.
2. Assign accumulator as 0.
3. Load the content of memory location specified into register.
4. Add content of memory location with accumulator and decrement register content by 01.
5. Check if register holds 00, if so store the value of accumulator in memory location.

**PROGRAM:**

LXI H,8000

XRA A

MOV B,M

LOOP: ADD M

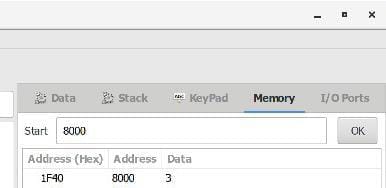
DCR B

JNZ LOOP

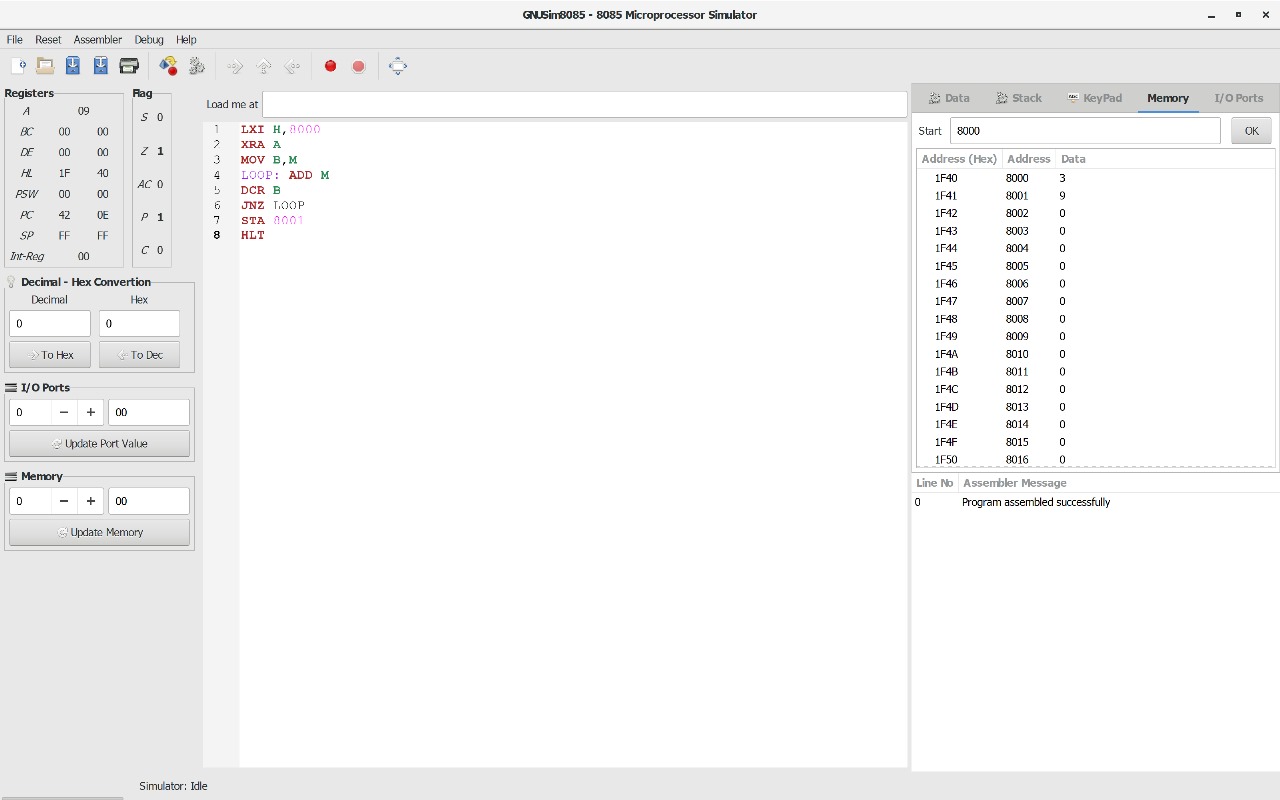
STA 8001

HLT

**INPUT:**



**OUTPUT:**

****

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**EXP NO: 17**

**AIM:** To compute one’s and two’s complement using 8085 processor.

**ALGORITHM:**

1. Load the base address of the array in a register pair.
2. Move the data from memory location into accumulator.
3. Convert all ones into zeros and zeros into ones.
4. Add 01 to the accumulator content.
5. Store the results of one’s and two’s complement.

**PROGRAM:**

LDA 3000

CMA

STA 3001

ADI 01

STA 3002

HLT

**INPUT:**

**OUTPUT:**

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**ROTATE LEFT OPERATION**

**EXP NO: 18**

**AIM:** To compute rotation of given data in left without carry using 8085 processor.

**ALGORITHM:**

1. Load the base address of the array in HL register pair.
2. Move the data from memory location into accumulator.
3. Shift left the accumulator content for four times.
4. Store the result in the specified location.

**PROGRAM:**

MVI A,02

RLC

RLC

RLC

RLC

STA 2000

HLT

**INPUT:**

**OUTPUT:**

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**ROTATE RIGHT OPERATION**

**EXP NO: 19**

**AIM:** To compute rotation of given data in right without carry using 8085 processor.

**ALGORITHM:**

1. Load the base address of the array in HL register pair.
2. Move the data from memory location into accumulator.
3. Shift right the accumulator content for four times left.
4. Store the result in the specified location.

**PROGRAM:**

MVI A,03

RRC

RRC

RRC

RRC

STA 2000

HLT

**INPUT:**

**OUTPUT:**

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.

**LOGICAL OPERATIONS**

**EXP NO: 20**

**AIM:** To compute various logical operations using 8085 processor.

**ALGORITHM:**

1. Load data to accumulator.
2. Load another data in register
3. Perform logical operations like AND, OR and XOR (Use ANA, ORA, XRA) with the accumulator content.
4. Store the result in specified memory location.

**PROGRAM:**

**AND OPERATION:**

MVI A,06

MVI B,04

ANA B

STA 2500

HLT

**OR OPERATION:**

MVI A,07

MVI B,06

ORA B

STA 2000

HLT

**XOR OPERATION:**

MVI A,03

MVI B,04

XRA B

STA 2000

HLT

**INPUT:**

**OUTPUT:**

**RESULT:** Thus the program was executed successfully using 8085 processor simulator.