NAME : PATEL MANAN VIJAYBHAI

CLASS : B.E. – III (SEM -5)

PRN NO. : 2020033800098747

SUBJECT : COMPUTER ORGANIZATION (CO)

EMAIL ID : mananpatel5921@gmail.com

CONTACT : +916354756653

# 1. ADDITION OF TWO 8 BITS NUMBER HAVING 16 BIT SUM

**AIM :** To perform addition of two 8 bit numbers using 8085.

## ALGORITHM :

1) Start the program by loading the first data into Accumulator.

2) Move the data to a register (B register).

3) Get the second data and load into Accumulator.

4) Add the two register contents.

5) Check for carry.

6) Store the value of sum and carry in memory location.

7) Terminate the program.

## PROGRAM :

```
# BEGIN 0000H
          MVI C,00       Initialize C register to 00
          LDA C050       Load the value to Accumulator
          MOV B,A        Move the content of Accumulator to B register.
          LDA C051       Load the value to Accumulator.
          ADD B          Add the value of register B to A
          JNC AHEAD      Jump on no carry.
          INR C          Increment value of register C

AHEAD:    STA C052       Store the value of Accumulator (SUM).
          MOV A,C        Move content of register C to Acc.
          STA C053       Store the value of Accumulator (CARRY)
          HLT            Halt the program.


# ORG C050
# DB 23H,9AH
```

## OBSERVATION :

**Input :** 23H (C050) -> 1st value

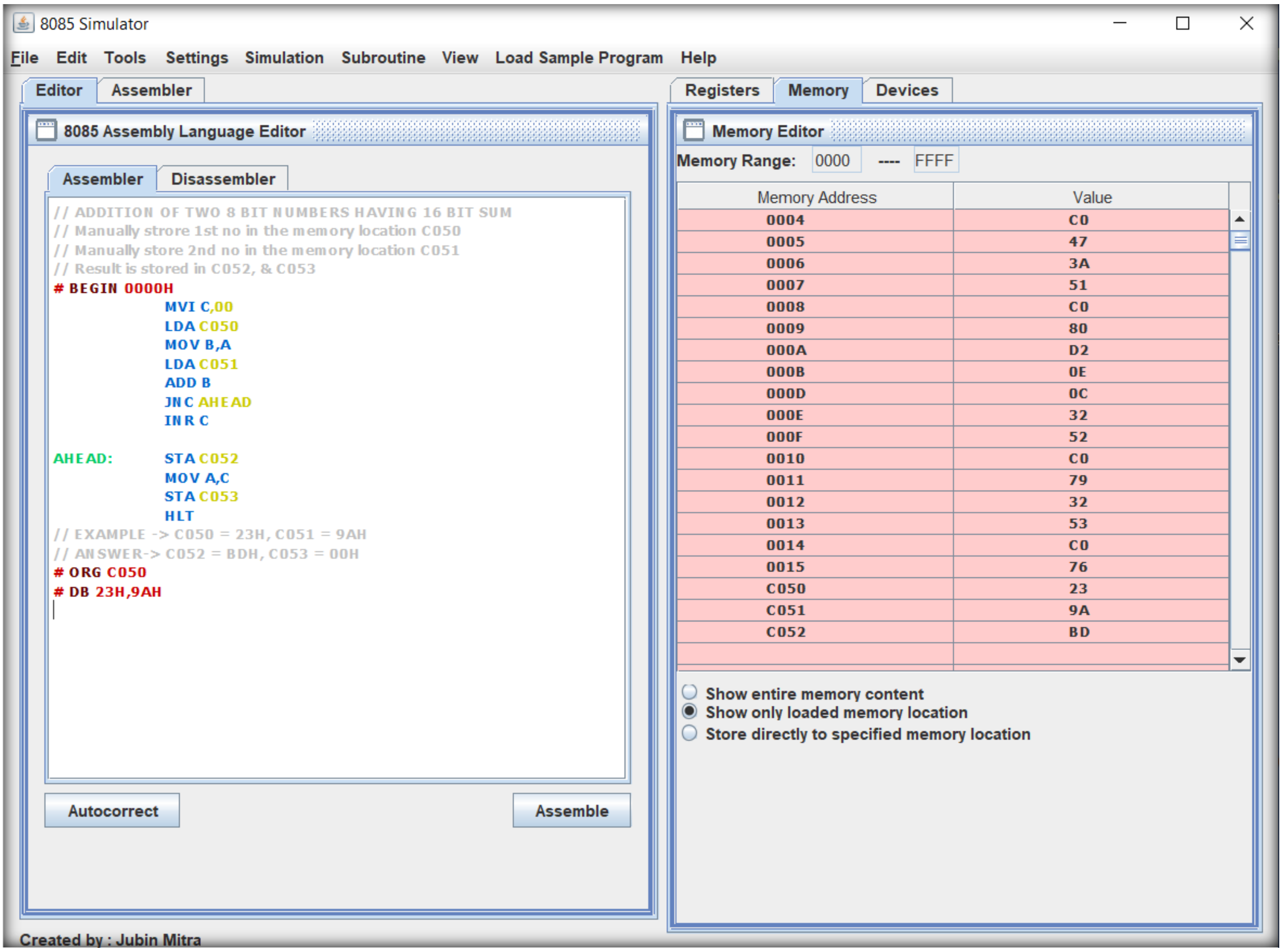
9AH (C051) -> 2nd value

**Output :** 00H (C053) -> Carry

BDH (C052) -> Sum

```
  0010 0011 -> 23H
  1001 1010 -> 9AH
-------------------
  1011 1101 -> BDH
```

**RESULT :** Thus the program to add two 8-bit numbers was executed.



```
// ADDITION OF TWO 8 BIT NUMBERS HAVING 16 BIT SUM
// Manually strore 1st no in the memory location C050
// Manually store 2nd no in the memory location C051
// Result is stored in C052, & C053
# BEGIN 0000H
            MVI C,00
            LDA C050
            MOV B,A
            LDA C051
            ADD B
            JNC AHEAD
            INR C

AHEAD:      STA C052
            MOV A,C
            STA C053
            HLT
// EXAMPLE -> C050 = 23H, C051 = 9AH
// ANSWER-> C052 = BDH, C053 = 00H
# ORG C050
# DB 23H,9AH
```

Memory Range: 0000 ---- FFFF

| Memory Address | Value |
|---|---|
| 0004 | C0 |
| 0005 | 47 |
| 0006 | 3A |
| 0007 | 51 |
| 0008 | C0 |
| 0009 | 80 |
| 000A | D2 |
| 000B | 0E |
| 000D | 0C |
| 000E | 32 |
| 000F | 52 |
| 0010 | C0 |
| 0011 | 79 |
| 0012 | 32 |
| 0013 | 53 |
| 0014 | C0 |
| 0015 | 76 |
| C050 | 23 |
| C051 | 9A |
| C052 | BD |

○ Show entire memory content
● Show only loaded memory location
○ Store directly to specified memory location

Created by : Jubin Mitra

# 2.SUBTRACTION OF TWO BIT NUMBERS

**AIM :** To perform the subtraction of two 8 bit numbers using 8085.

## ALGORITHM:

1) Start the program by loading the first data into Accumulator.
2) Move the data to a register (B register).
3) Get the second data and load into Accumulator.
4) Subtract the two register contents.
5) Check for carry.
6) If carry is present take 2's complement of Accumulator.
7) Store the value of borrow in memory location.
8) Store the difference value (present in Accumulator) to a memory
9) location and terminate the program.

## PROGRAM :

```
# BEGIN 0000H
        MVI C,00      Initialize C to 00
        LDA C050      Load the value to Acc.
        MOV B,A       Move the content of Acc to B register.
        LDA C051      Load the value to Acc.
        SUB B
        JNC AHEAD     Jump on no carry.
        CMA           Complement Accumulator contents.
        INR A         Increment value in Accumulator.
        INR C         Increment value in register C
AHEAD:  STA C052      Store the value of A-reg to memory address.
        MOV A,C       Move contents of register C to Accumulator.
        STA C053      Store the value of Accumulator to memory address.
        HLT           Halt the program.
```

**CASE 1 :** When minuend is greater than subtrahend.
# ORG C050
# DB 06H,02H

```
        0000 0110 -> 06H
        0000 0010 -> 02H
-----------------------------------------------------------------
        0000 0100 -> 04H   (C053 = 01H  >> No Borrow)
```

**Input :** 06H (C050)
02H (C051)

**Output :** 04H (C052) -> Difference
01H (C053) -> No Borrow

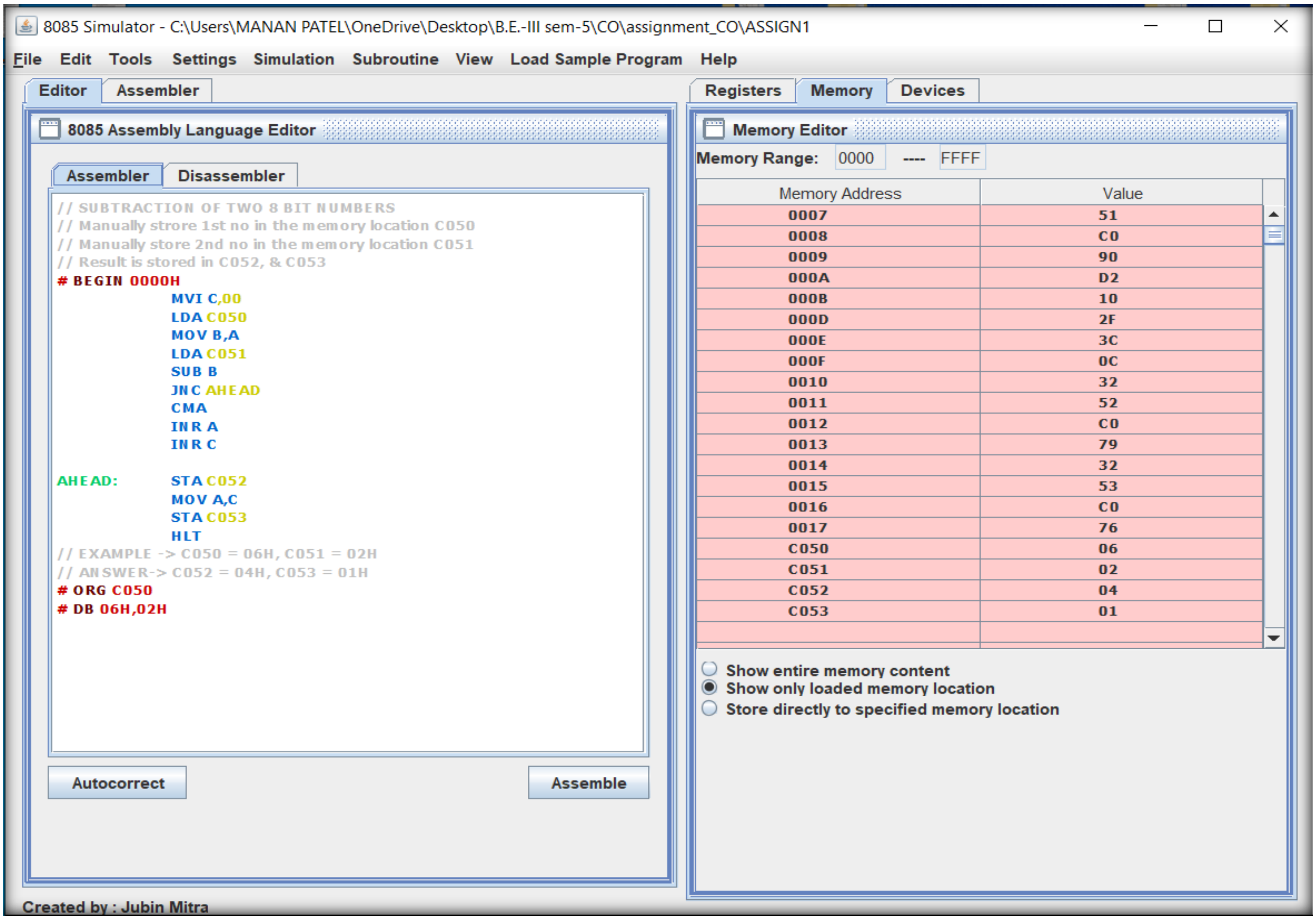
**CASE 2 :** When minuend is less than subtrahend.
# ORG C050
# DB 02H,06H

```
        0000 0010 -> 02H
        0000 0110 -> 06H
-----------------------------------------------------------------
        0000 0100 -> 04H   (C053 = 00H  >> Borrow)
```

**Input :** 02H (C050)
06H (C051)

**Output :** 04H (C052) -> Difference
00H (C053) -> Borrow

**RESULT :** Thus the program to subtract two 8-bit numbers was executed.



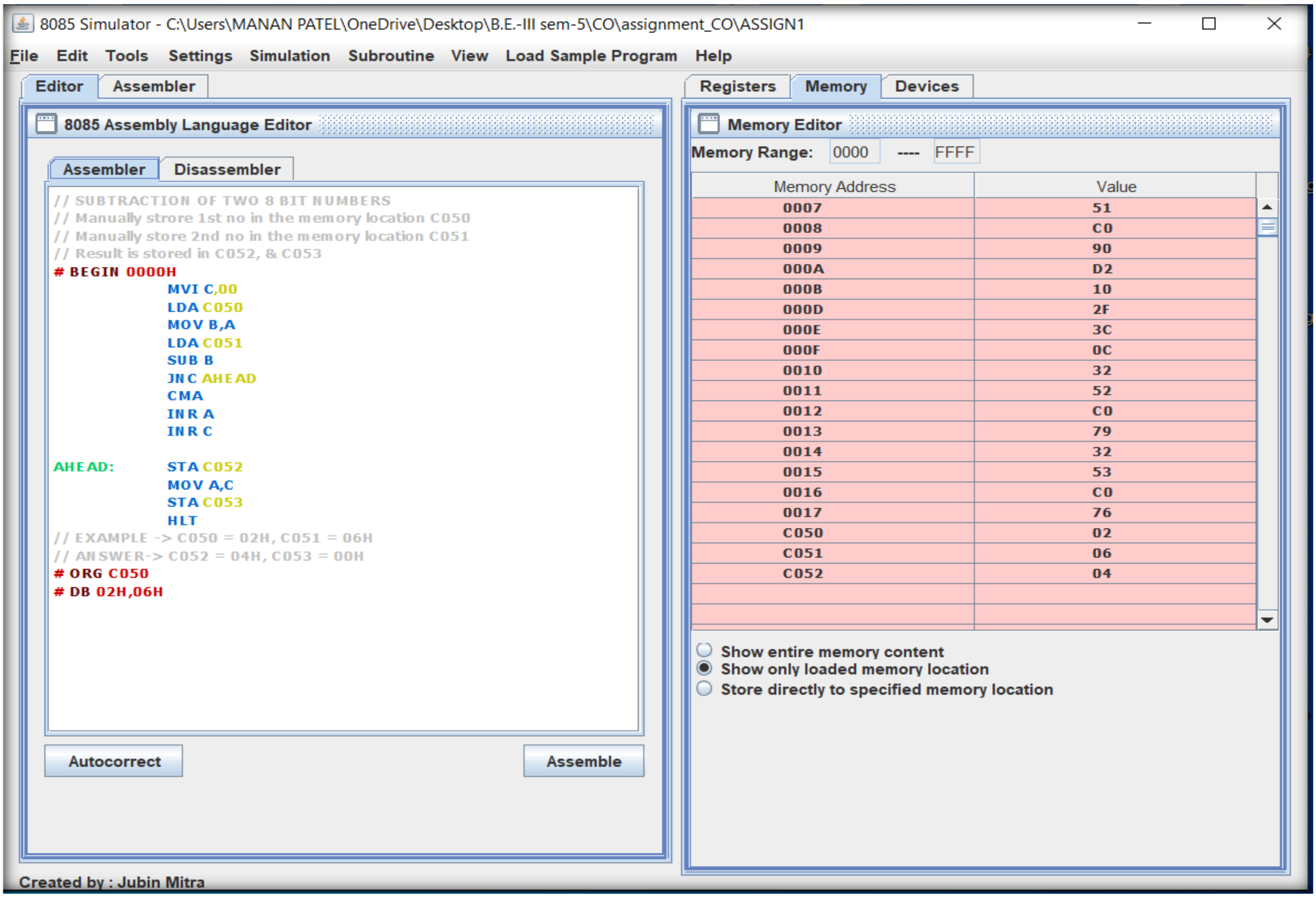
```
// SUBTRACTION OF TWO 8 BIT NUMBERS
// Manually strore 1st no in the memory location C050
// Manually store 2nd no in the memory location C051
// Result is stored in C052, & C053
# BEGIN 0000H
            MVI C,00
            LDA C050
            MOV B,A
            LDA C051
            SUB B
            JNC AHEAD
            CMA
            INR A
            INR C

AHEAD:      STA C052
            MOV A,C
            STA C053
            HLT
// EXAMPLE -> C050 = 06H, C051 = 02H
// ANSWER-> C052 = 04H, C053 = 01H
# ORG C050
# DB 06H,02H
```

Memory Range: 0000 ---- FFFF

| Memory Address | Value |
|---|---|
| 0007 | 51 |
| 0008 | C0 |
| 0009 | 90 |
| 000A | D2 |
| 000B | 10 |
| 000D | 2F |
| 000E | 3C |
| 000F | 0C |
| 0010 | 32 |
| 0011 | 52 |
| 0012 | C0 |
| 0013 | 79 |
| 0014 | 32 |
| 0015 | 53 |
| 0016 | C0 |
| 0017 | 76 |
| C050 | 06 |
| C051 | 02 |
| C052 | 04 |
| C053 | 01 |

○ Show entire memory content
● Show only loaded memory location
○ Store directly to specified memory location



```
// SUBTRACTION OF TWO 8 BIT NUMBERS
// Manually strore 1st no in the memory location C050
// Manually store 2nd no in the memory location C051
// Result is stored in C052, & C053
# BEGIN 0000H
            MVI C,00
            LDA C050
            MOV B,A
            LDA C051
            SUB B
            JNC AHEAD
            CMA
            INR A
            INR C

AHEAD:      STA C052
            MOV A,C
            STA C053
            HLT
// EXAMPLE -> C050 = 02H, C051 = 06H
// ANSWER-> C052 = 04H, C053 = 00H
# ORG C050
# DB 02H,06H
```

Memory Range: 0000 ---- FFFF

| Memory Address | Value |
|---|---|
| 0007 | 51 |
| 0008 | C0 |
| 0009 | 90 |
| 000A | D2 |
| 000B | 10 |
| 000D | 2F |
| 000E | 3C |
| 000F | 0C |
| 0010 | 32 |
| 0011 | 52 |
| 0012 | C0 |
| 0013 | 79 |
| 0014 | 32 |
| 0015 | 53 |
| 0016 | C0 |
| 0017 | 76 |
| C050 | 02 |
| C051 | 06 |
| C052 | 04 |

○ Show entire memory content
● Show only loaded memory location
○ Store directly to specified memory location

# 3.MULTIPLICATION OF TWO 8 BIT NUMBERS

**AIM:** To perform the multiplication of two 8 bit numbers using 8085.

## ALGORITHM:

1) Start the program by loading HL register pair with address of memory location.

2) Move the data to a register (B register).

3) Get the second data and load into Accumulator.

4) Add the two register contents.

5) Check for carry.

6) Increment the value of carry.

7) Check whether repeated addition is over and store the value of product and carry in memory location.

8) Terminate the program.

## PROGRAM:

```
# BEGIN 0000H
        MVI D,00      Initialize register D to 00
        MVI A,00      Initialize Accumulator content to 00
        LXI H,C050
        MOV B,M       Get the first number in B – reg
        INX H
        MOV C,M       Get the second number in C- reg.

LOOP:   ADD B         Add content of A - reg to register B.
        JNC AHEAD     Jump on no carry to AHEAD.
        INR D         Increment content of register D

AHEAD:  DCR C         Decrement content of register C.
        JNZ LOOP      Jump on no zero to address
        STA C052      Store the result in Memory
        MOV A,D
        STA C053      Store the MSB of result in Memory
        HLT           Terminate the program.

# ORG C050
# DB FFH,02H
```

## OBSERVATION:

**Input:** FFH (C050)

02H (C051)

**Output:** FEH (C052) ->Multiplication

01H (C053) -> MSB

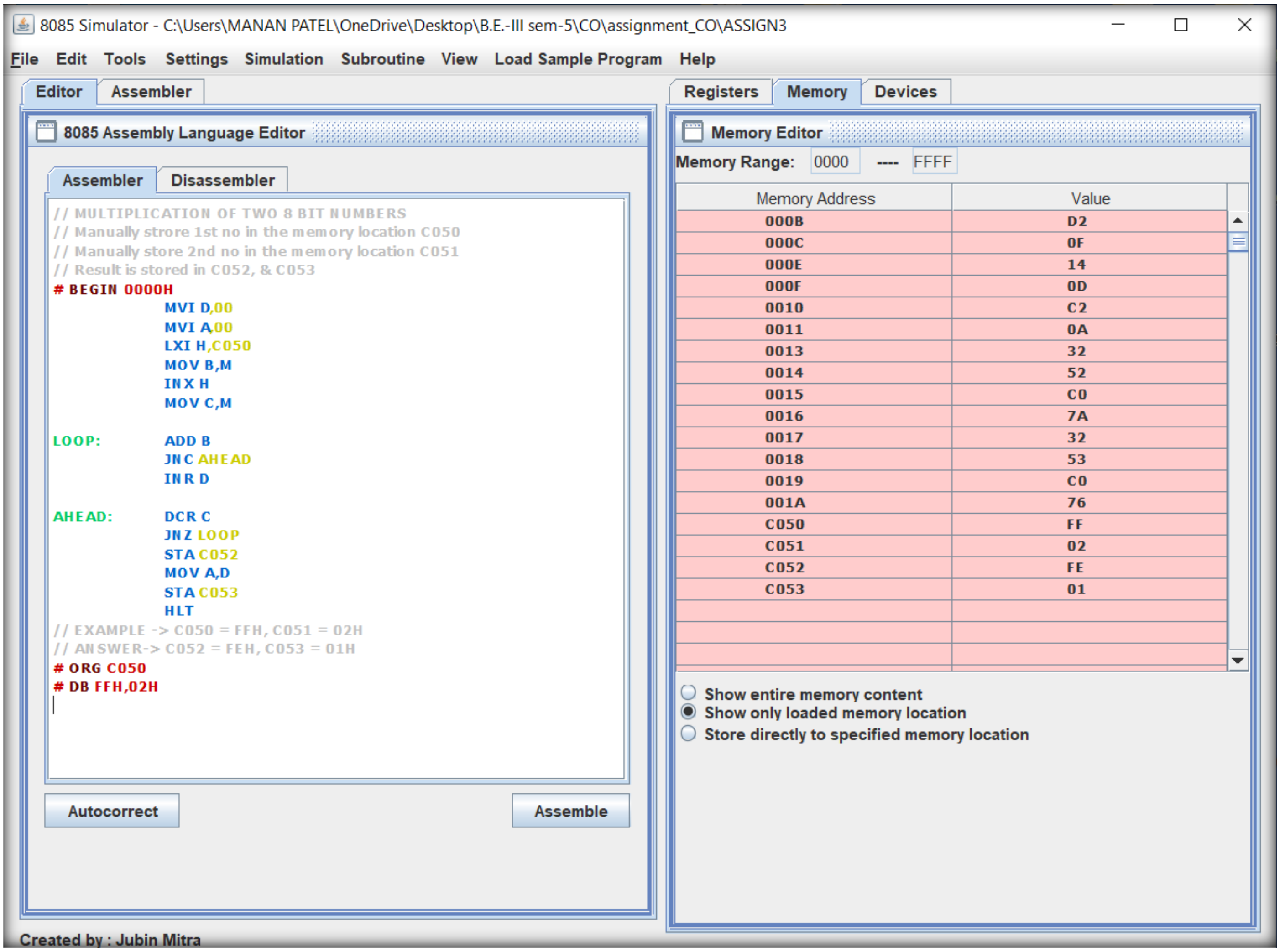
1111 1111 -> FFH

1111 1111 -> FFH

---

1 1111 1110 -> FEH (MSB = 01H)

**RESULT:** Thus the program to multiply two 8-bit numbers was executed.


8085 Simulator - C:\Users\MANAN PATEL\OneDrive\Desktop\B.E.-III sem-5\CO\assignment_CO\ASSIGN3
File Edit Tools Settings Simulation Subroutine View Load Sample Program Help
Editor Assembler
8085 Assembly Language Editor
Assembler Disassembler
Registers Memory Devices
Memory Editor
Autocorrect
Assemble
Created by : Jubin Mitra

```
// MULTIPLICATION OF TWO 8 BIT NUMBERS
// Manually strore 1st no in the memory location C050
// Manually store 2nd no in the memory location C051
// Result is stored in C052, & C053
# BEGIN 0000H
                MVI D,00
                MVI A,00
                LXI H,C050
                MOV B,M
                INX H
                MOV C,M

LOOP:           ADD B
                JNC AHEAD
                INR D

AHEAD:          DCR C
                JNZ LOOP
                STA C052
                MOV A,D
                STA C053
                HLT
// EXAMPLE -> C050 = FFH, C051 = 02H
// ANSWER-> C052 = FEH, C053 = 01H
# ORG C050
# DB FFH,02H
```

Memory Range: 0000 ---- FFFF

| Memory Address | Value |
|---|---|
| 000B | D2 |
| 000C | 0F |
| 000E | 14 |
| 000F | 0D |
| 0010 | C2 |
| 0011 | 0A |
| 0013 | 32 |
| 0014 | 52 |
| 0015 | C0 |
| 0016 | 7A |
| 0017 | 32 |
| 0018 | 53 |
| 0019 | C0 |
| 001A | 76 |
| C050 | FF |
| C051 | 02 |
| C052 | FE |
| C053 | 01 |

○ Show entire memory content
● Show only loaded memory location
○ Store directly to specified memory location

# 4.DIVISION OF TWO 8 BIT NUMBERS

**AIM:** To perform the division of two 8 bit numbers using 8085.

## ALGORITHM:

1) Start the program by loading HL register pair with address of memory location.

2) Move the data to a register(B register).

3) Get the second data and load into Accumulator.

4) Compare the two numbers to check for carry.

5) Subtract the two numbers.

6) Increment the value of carry .

7) Check whether repeated subtraction is over and store the value of product and carry in memory location.

8) Terminate the program.

## PROGRAM:

```
# BEGIN 0000H
        LXI H,C050
        MOV B,M      Get the dividend in B – reg.
        MVI C,00     Clear C – reg for qoutient
        INX H
        MOV A,M      Get the divisor in A – reg.

AHEAD:  CMP B        Compare A - reg with register B.
        JC LOOP      Jump on carry to LOOP
        SUB B        Subtract A – reg from B- reg.
        INR C        Increment content of register C.
        JMP AHEAD    Jump to AHEAD

LOOP:   STA C052     Store the remainder in Memory
        MOV A,C
        STA C053     Store the quotient in memory
        HLT          Terminate the program.


# ORG C050
# DB FFH,FFH
```

## OBSERVATION:

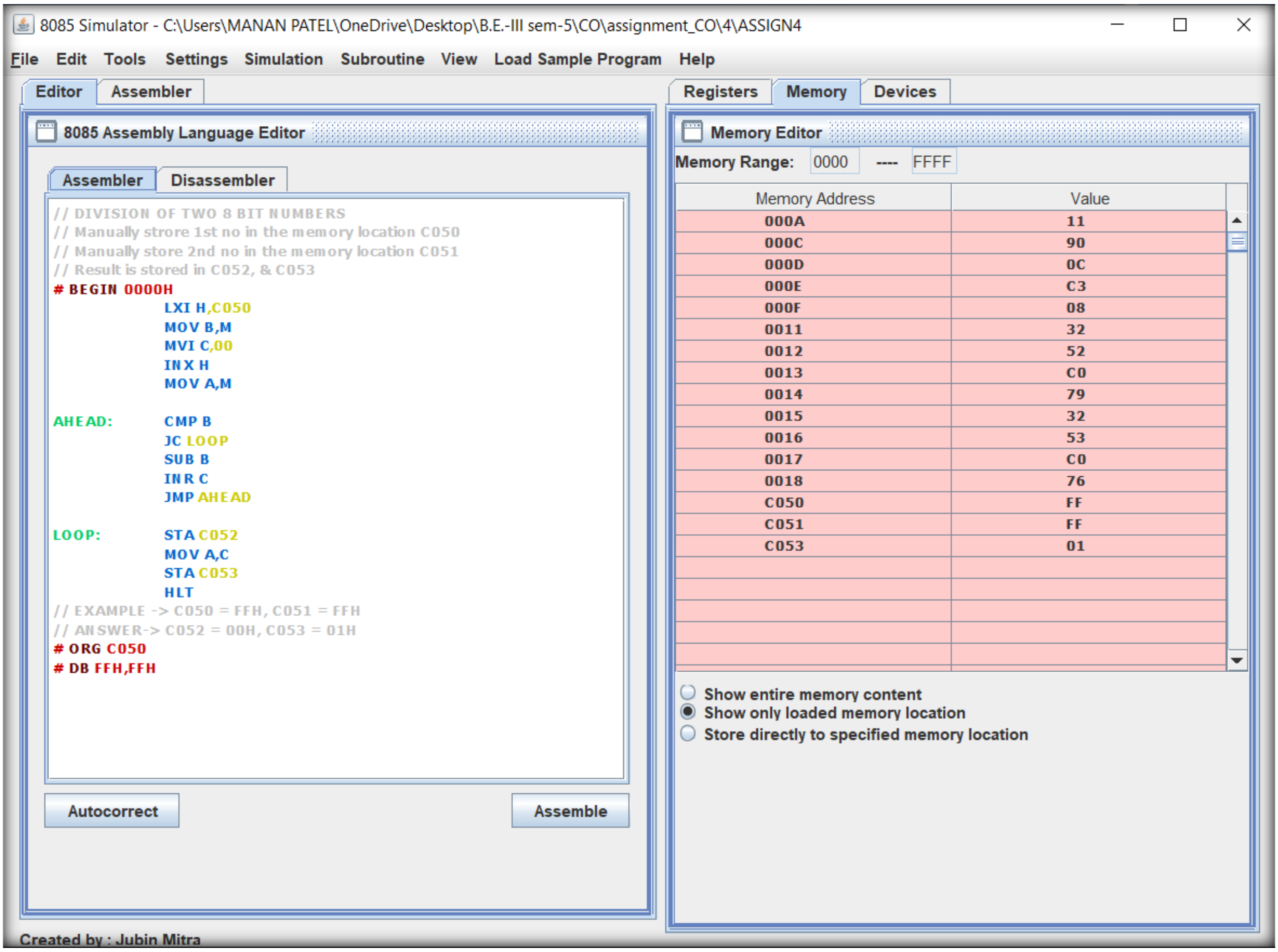
**Input:** FF (C050)

FF (C051)

**Output:** 00H (C052) ---- Remainder

01H (C053) ---- Quotient

**RESULT :** Thus the program to divide two 8-bit numbers was executed.



8085 Simulator - C:\Users\MANAN PATEL\OneDrive\Desktop\B.E.-III sem-5\CO\assignment_CO\4\ASSIGN4
File Edit Tools Settings Simulation Subroutine View Load Sample Program Help
Editor Assembler
Registers Memory Devices
8085 Assembly Language Editor
Assembler Disassembler
Memory Editor

```
// DIVISION OF TWO 8 BIT NUMBERS
// Manually strore 1st no in the memory location C050
// Manually store 2nd no in the memory location C051
// Result is stored in C052, & C053
# BEGIN 0000H
                LXI H,C050
                MOV B,M
                MVI C,00
                INX H
                MOV A,M

AHEAD:          CMP B
                JC LOOP
                SUB B
                INR C
                JMP AHEAD

LOOP:           STA C052
                MOV A,C
                STA C053
                HLT
// EXAMPLE -> C050 = FFH, C051 = FFH
// ANSWER-> C052 = 00H, C053 = 01H
# ORG C050
# DB FFH,FFH
```

Memory Range: 0000 ---- FFFF

| Memory Address | Value |
|---|---|
| 000A | 11 |
| 000C | 90 |
| 000D | 0C |
| 000E | C3 |
| 000F | 08 |
| 0011 | 32 |
| 0012 | 52 |
| 0013 | C0 |
| 0014 | 79 |
| 0015 | 32 |
| 0016 | 53 |
| 0017 | C0 |
| 0018 | 76 |
| C050 | FF |
| C051 | FF |
| C053 | 01 |

○ Show entire memory content
● Show only loaded memory location
○ Store directly to specified memory location

# 5.WRITE A 8085 PROGRAM TO FIND LARGEST NUMBER IN THE GIVEN ARRAY OF NUMBERS

**AIM:** To find the largest number in an array of data using 8085 instruction set.

## ALGORITHM:

1) Load the address of the first element of the array in HL pair

2) Move the count to B – reg.

3) Increment the pointer

4) Get the first data in A – reg.

5) Decrement the count.

6) Increment the pointer

7) Compare the content of memory addressed by HL pair with that of A - reg.

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 – reg.

10) Decrement the count

11) Check for Zero of the count. If ZF = 0, go to step 6, or if ZF = 1 go to next step.

12) Store the largest data in memory.

13) Terminate the program.

## PROGRAM:

```
# BEGIN 0000H
           LXI H,C050     Set pointer for array
           MOV B,M        Load the Count
           INX H
           MOV A,M        Set 1st element as largest data
           DCR B          Decrement the count
```

```
LOOP:     INX H
          CMP M        If A- reg > M go to AHEAD
          JNC AHEAD
          MOV A,M      Set the new value as largest

AHEAD:    DCR B
          JNZ LOOP     Repeat comparisons till count = 0
          STA C150     Store the largest value at  C150
          HLT


# ORG C050
# DB 05H  ,   9AH,F1H,1FH,26H,FEH
```

## OBSERVATION:

**Input :** 05H (C050) ----- Array Size

9AH (C051)

F1H (C052)

1FH (C053)

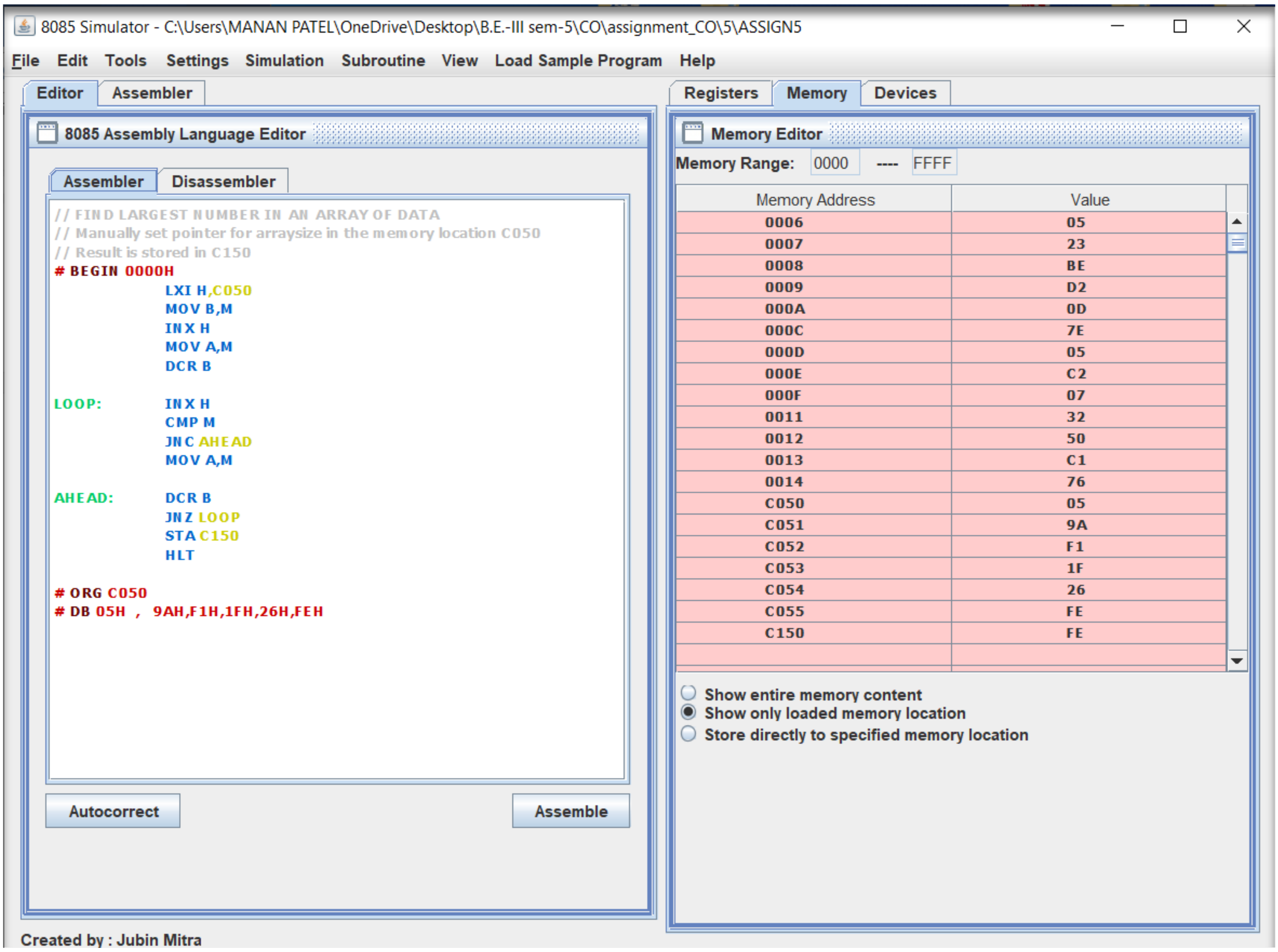
26H (C054)

FEH (C055)

**Output:**

FEH (C150)

**RESULT:** Thus the program to find the largest number in an array of data was executed.



```
// FIND LARGEST NUMBER IN AN ARRAY OF DATA
// Manually set pointer for arraysize in the memory location C050
// Result is stored in C150
# BEGIN 0000H
                LXI H,C050
                MOV B,M
                INX H
                MOV A,M
                DCR B

LOOP:           INX H
                CMP M
                JNC AHEAD
                MOV A,M

AHEAD:          DCR B
                JNZ LOOP
                STA C150
                HLT

# ORG C050
# DB 05H ,  9AH,F1H,1FH,26H,FEH
```

Memory Range: 0000 ---- FFFF

| Memory Address | Value |
|---|---|
| 0006 | 05 |
| 0007 | 23 |
| 0008 | BE |
| 0009 | D2 |
| 000A | 0D |
| 000C | 7E |
| 000D | 05 |
| 000E | C2 |
| 000F | 07 |
| 0011 | 32 |
| 0012 | 50 |
| 0013 | C1 |
| 0014 | 76 |
| C050 | 05 |
| C051 | 9A |
| C052 | F1 |
| C053 | 1F |
| C054 | 26 |
| C055 | FE |
| C150 | FE |

○ Show entire memory content
● Show only loaded memory location
○ Store directly to specified memory location

# 6.WRITE A 8085 PROGRAM TO FIND SMALLEST NUMBER IN THE GIVEN ARRAY OF NUMBERS

**AIM :** To find the smallest number in an array of data using 8085 instruction set.

**ALGORITHM:**

1) Load the address of the first element of the array in HL pair

2) Move the count to B – reg.

3) Increment the pointer

4) Get the first data in A – reg.

5) Decrement the count.

6) Increment the pointer

7) Compare the content of memory addressed by HL pair with that of A - reg.

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 – reg.

10) Decrement the count

11) Check for Zero of the count. If ZF = 0, go to step 6, or if ZF = 1 go to next step.

12) Store the smallest data in memory.

13) Terminate the program.

## PROGRAM :

```
# BEGIN 0000H
        LXI H,C050      Set pointer for array
        MOV B,M         Load the Count
        INX H
        MOV A,M         Set 1st element as largest data
        DCR B           Decrement the count

LOOP:   INX H
        CMP M           If A- reg < M go to AHEAD
        JC AHEAD
        MOV A,M         Set the new value as smallest

AHEAD:  DCR B
        JNZ LOOP        Repeat comparisons till count = 0
        STA C150        Store the largest value at C150
        HLT


# ORG C050
# DB 05H , 9AH,F1H,1FH,26H,FEH
```

## OBSERVATION :

**Input:** 05H (C050) ----- Array Size

9AH (C051)

F1H (C052)

1FH (C053)

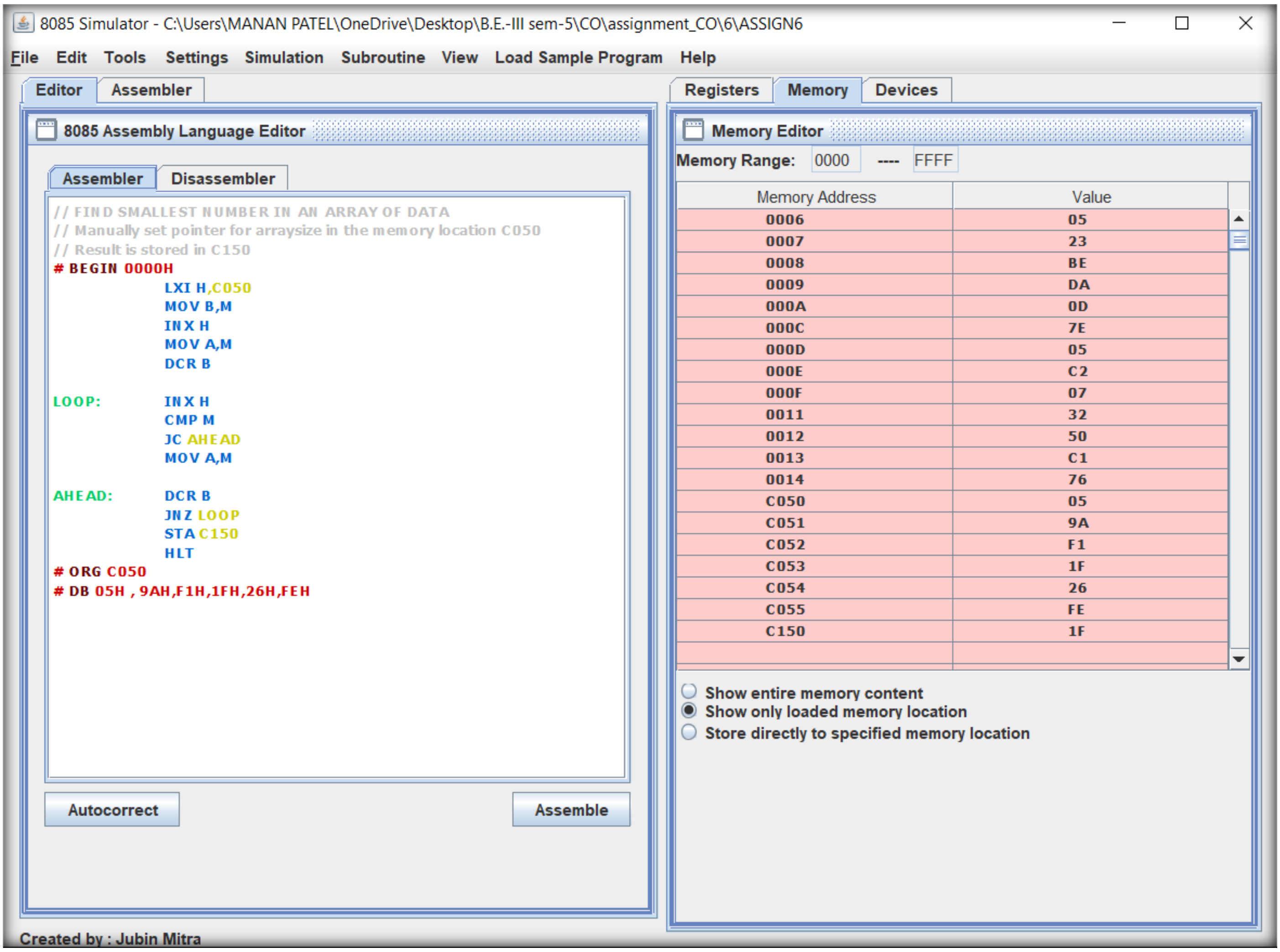
26H (C054)

FEH (C055)

**Output:**

1FH (C150)

**RESULT:** Thus the program to find the smallest number in an array of data was executed.


8085 Simulator - C:\Users\MANAN PATEL\OneDrive\Desktop\B.E.-III sem-5\CO\assignment_CO\6\ASSIGN6
File Edit Tools Settings Simulation Subroutine View Load Sample Program Help
Editor Assembler
8085 Assembly Language Editor
Assembler Disassembler
Registers Memory Devices
Memory Editor
Autocorrect
Assemble
Created by : Jubin Mitra

```
// FIND SMALLEST NUMBER IN AN ARRAY OF DATA
// Manually set pointer for arraysize in the memory location C050
// Result is stored in C150
# BEGIN 0000H
                LXI H,C050
                MOV B,M
                INX H
                MOV A,M
                DCR B

LOOP:           INX H
                CMP M
                JC AHEAD
                MOV A,M

AHEAD:          DCR B
                JNZ LOOP
                STA C150
                HLT
# ORG C050
# DB 05H , 9AH,F1H,1FH,26H,FEH
```

Memory Range: 0000 ---- FFFF

| Memory Address | Value |
|---|---|
| 0006 | 05 |
| 0007 | 23 |
| 0008 | BE |
| 0009 | DA |
| 000A | 0D |
| 000C | 7E |
| 000D | 05 |
| 000E | C2 |
| 000F | 07 |
| 0011 | 32 |
| 0012 | 50 |
| 0013 | C1 |
| 0014 | 76 |
| C050 | 05 |
| C051 | 9A |
| C052 | F1 |
| C053 | 1F |
| C054 | 26 |
| C055 | FE |
| C150 | 1F |

○ Show entire memory content
● Show only loaded memory location
○ Store directly to specified memory location

# 7.WRITE A 8085 PROGRAM TO SORT THE GIVEN ARRAY OF NUMBERS IN ASCENDING ORDER

**AIM :** To write a program to arrange an array of data in ascending order

**ALGORITHM:**

1) Initialize HL pair as memory pointer
2) Get the count at 4200 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 steps 5 and 7 till the value in D- register become zero
9) Decrement C –register content by 1
10) Repeat steps 3 to 9 till the value in C – register becomes zero

## PROGRAM:

```
# BEGIN 0000H
            LXI H,C050
            MOV C,M
            DCR C

REPEAT:             MOV D,C
            LXI H,C051

LOOP:       MOV A,M
            INX H
            CMP M
            JC SKIP
            MOV B,M
            MOV M,A
            DCX H
            MOV M,B
            INX H

SKIP:       DCR D
            JNZ LOOP
            DCR C
            JNZ REPEAT
            HLT

# ORG C050
# DB 05H , 9AH,F1H,1FH,26H,FEH
```

## OBSERVATION :

**Input:** 05H (C050) →(Array Size)

9AH (C051)

F1H (C052)

1FH (C053)

26H (C054)

FEH (C055)

**Output:** 05H (C050) →(Array Size)

1FH (C051)

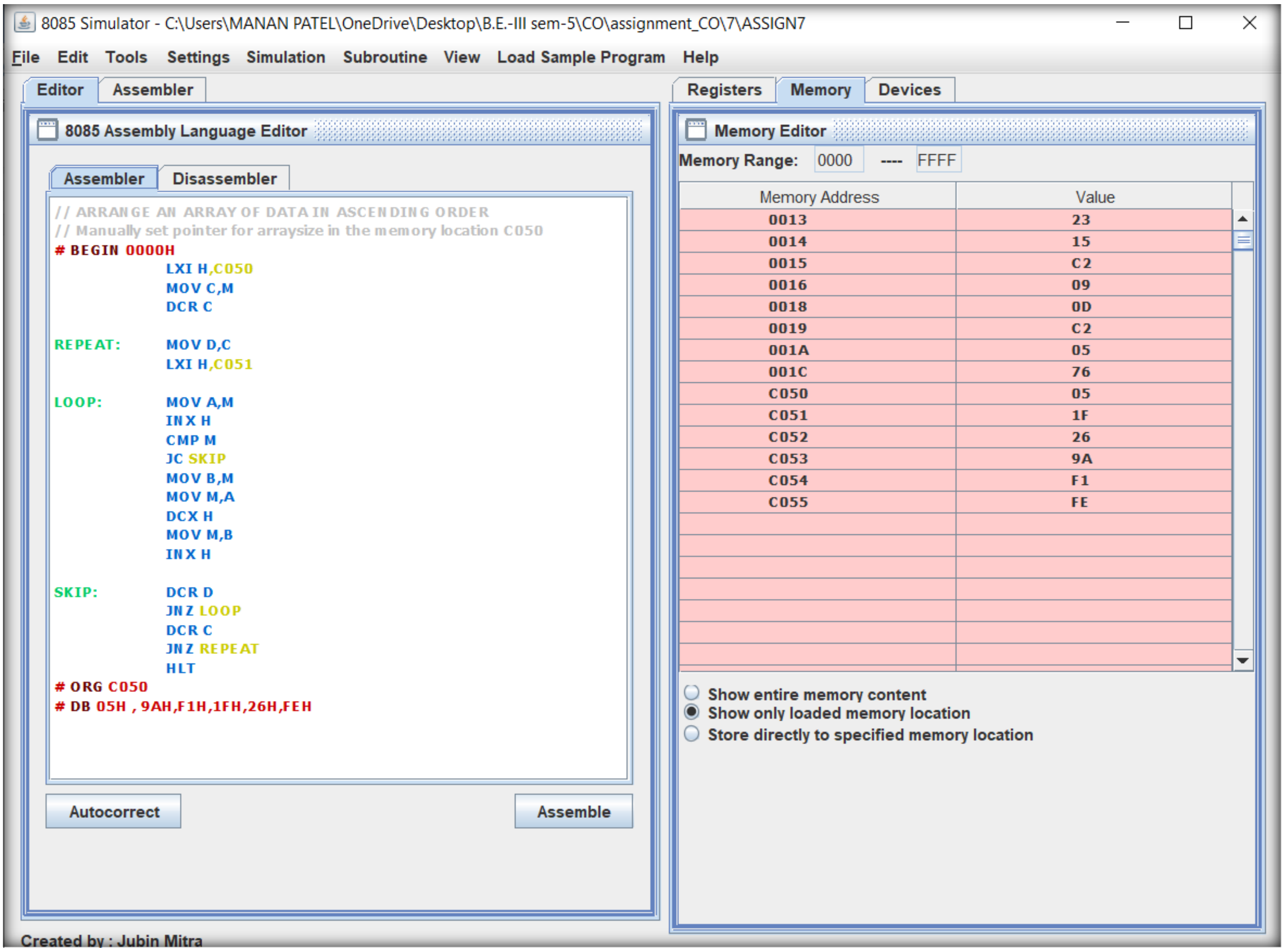
26H (C052)

9AH (C053)

F1H (C054)

FEH (C055)

**RESULT :** Thus the given array of data was arranged in ascending order.


8085 Simulator - C:\Users\MANAN PATEL\OneDrive\Desktop\B.E.-III sem-5\CO\assignment_CO\7\ASSIGN7
File Edit Tools Settings Simulation Subroutine View Load Sample Program Help
Editor
Assembler
8085 Assembly Language Editor
Assembler
Disassembler
Autocorrect
Assemble
Registers
Memory
Devices
Memory Editor
Created by : Jubin Mitra

```
// ARRANGE AN ARRAY OF DATA IN ASCENDING ORDER
// Manually set pointer for arraysize in the memory location C050
# BEGIN 0000H
                LXI H,C050
                MOV C,M
                DCR C

REPEAT:         MOV D,C
                LXI H,C051

LOOP:           MOV A,M
                INX H
                CMP M
                JC SKIP
                MOV B,M
                MOV M,A
                DCX H
                MOV M,B
                INX H

SKIP:           DCR D
                JNZ LOOP
                DCR C
                JNZ REPEAT
                HLT
# ORG C050
# DB 05H , 9AH,F1H,1FH,26H,FEH
```

Memory Range: 0000 ---- FFFF

| Memory Address | Value |
|---|---|
| 0013 | 23 |
| 0014 | 15 |
| 0015 | C2 |
| 0016 | 09 |
| 0018 | 0D |
| 0019 | C2 |
| 001A | 05 |
| 001C | 76 |
| C050 | 05 |
| C051 | 1F |
| C052 | 26 |
| C053 | 9A |
| C054 | F1 |
| C055 | FE |

○ Show entire memory content
● Show only loaded memory location
○ Store directly to specified memory location

# 8.WRITE A 8085 PROGRAM TO SORT THE GIVEN ARRAY OF NUMBERS IN DESCENDING ORDER

AIM : To write a program to arrange an array of data in descending order

## ALGORITHM :

1) Initialize HL pair as memory pointer
2) Get the count at 4200 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 steps 5 and 7 till the value in D- register become zero
9) Decrement C –register content by 1
10) Repeat steps 3 to 9 till the value in C – register becomes zero

## PROGRAM :

```
# BEGIN 0000H
          LXI H,C050
          MOV C,M
          DCR C

REPEAT:   MOV D,C
          LXI H,C051

LOOP:     MOV A,M
          INX H
          CMP M
          JNC SKIP
          MOV B,M
          MOV M,A
          DCX H
```

```
         MOV M,B
         INX H

SKIP:    DCR D
         JNZ LOOP
         DCR C
         JNZ REPEAT
         HLT

# ORG C050
# DB 05H , 9AH,F1H,1FH,26H,FEH
```

## OBSERVATION :

**Input:** 05H (C050) →(Array Size)

9AH (C051)

F1H (C052)

1FH (C053)

26H (C054)

FEH (C055)

**Output:** 05H (C050) →(Array Size)

FEH (C051)

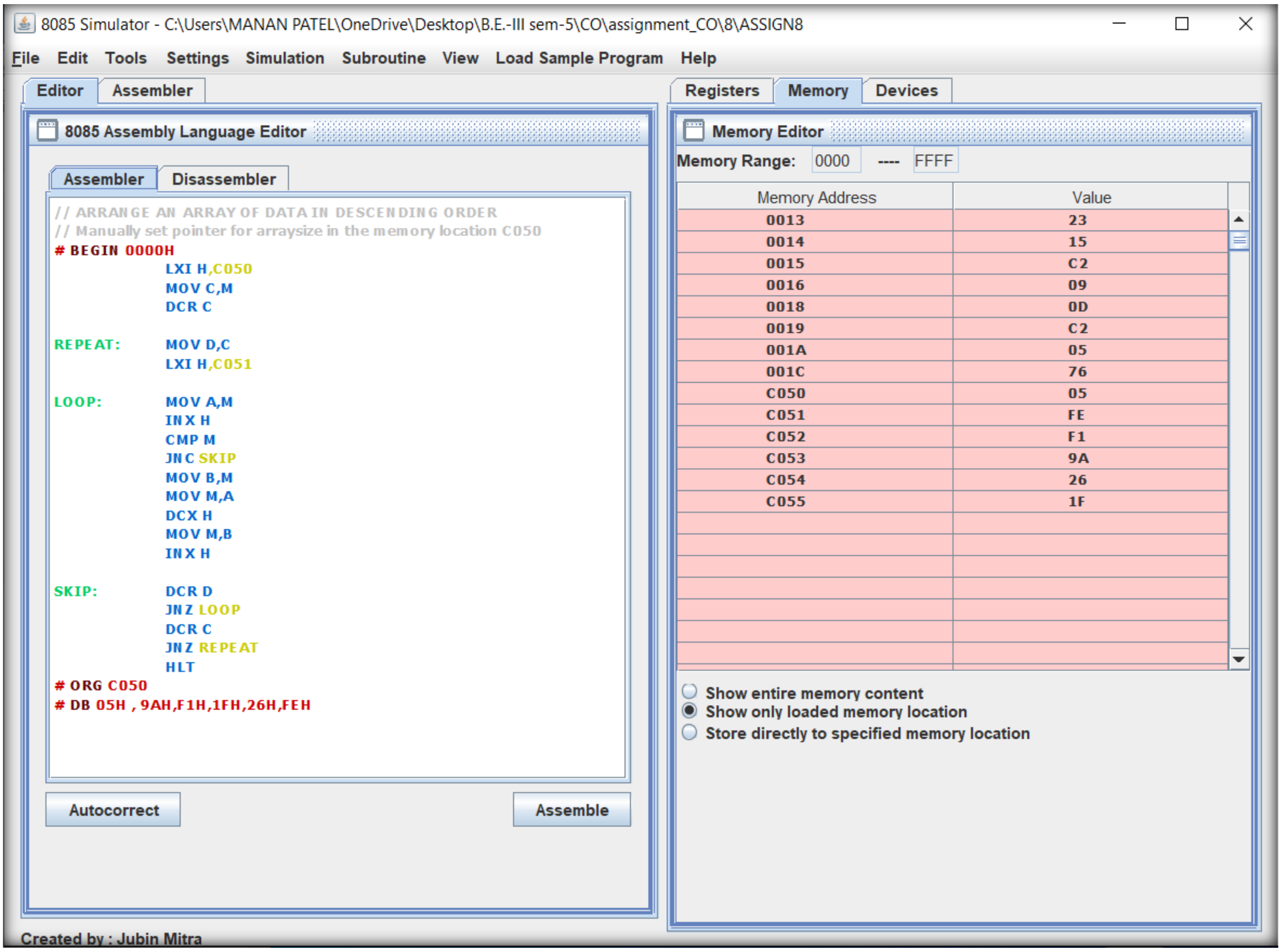
F1H (C052)

9AH (C053)

26H (C054)

1FH (C055)

**RESULT :** Thus the given array of data was arranged in descending order.


8085 Simulator - C:\Users\MANAN PATEL\OneDrive\Desktop\B.E.-III sem-5\CO\assignment_CO\8\ASSIGN8
File Edit Tools Settings Simulation Subroutine View Load Sample Program Help
Editor Assembler
8085 Assembly Language Editor
Assembler Disassembler
Autocorrect
Assemble
Registers Memory Devices
Memory Editor
Created by : Jubin Mitra

```
// ARRANGE AN ARRAY OF DATA IN DESCENDING ORDER
// Manually set pointer for arraysize in the memory location C050
# BEGIN 0000H
                LXI H,C050
                MOV C,M
                DCR C

REPEAT:         MOV D,C
                LXI H,C051

LOOP:           MOV A,M
                INX H
                CMP M
                JNC SKIP
                MOV B,M
                MOV M,A
                DCX H
                MOV M,B
                INX H

SKIP:           DCR D
                JNZ LOOP
                DCR C
                JNZ REPEAT
                HLT
# ORG C050
# DB 05H , 9AH,F1H,1FH,26H,FEH
```

Memory Range: 0000 ---- FFFF

| Memory Address | Value |
|---|---|
| 0013 | 23 |
| 0014 | 15 |
| 0015 | C2 |
| 0016 | 09 |
| 0018 | 0D |
| 0019 | C2 |
| 001A | 05 |
| 001C | 76 |
| C050 | 05 |
| C051 | FE |
| C052 | F1 |
| C053 | 9A |
| C054 | 26 |
| C055 | 1F |

○ Show entire memory content
● Show only loaded memory location
○ Store directly to specified memory location

# 9.WRITE A 8085 PROGRAM TO CONVERT A GIVEN HEXADECIMAL NUMBER TO BCD NUMBER

**AIM :** To convert two BCD numbers in memory to the equivalent HEX number using 8085 instruction set

**ALGORITHM :**

1) Initialize memory pointer to 4150 H
2) Get the Most Significant Digit (MSD)
3) Multiply the MSD by ten using repeated addition
4) Add the Least Significant Digit (LSD) to the result obtained in previous step
5) Store the HEX data in Memory

## PROGRAM :

```
# BEGIN 0000H
        LXI H,C050
        MOV A,M      Initialize memory pointer
        ADD A        MSD X 2
        MOV B,A      Store MSD X 2
        ADD A        MSD X 4
        ADD A        MSD X 8
        ADD B        MSD X 10
        INX H        Point to LSD
        ADD M        Add to form HEX
        INX H
        ADD M
        INX H
        MOV M,A      Store the result
        HLT


# ORG C050
# DB 02,09
```

**OBSERVATION :**

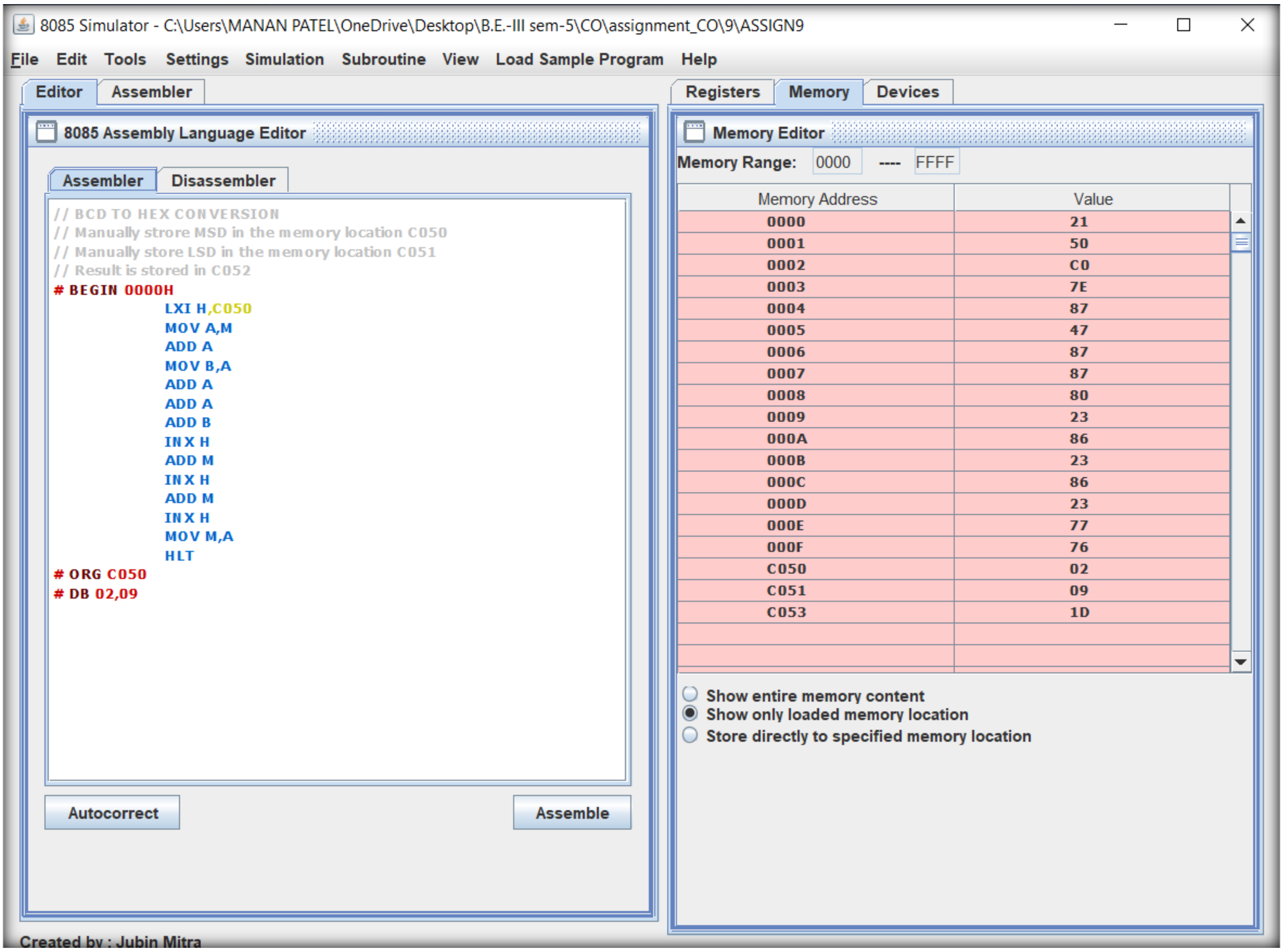
**Input :** 02 (C050) -> MSD

09 (C051) -> LSD

**Output :**

1DH (C053)

**RESULT :** Thus the program to convert BCD data to HEX data was executed.


8085 Simulator - C:\Users\MANAN PATEL\OneDrive\Desktop\B.E.-III sem-5\CO\assignment_CO\9\ASSIGN9
File Edit Tools Settings Simulation Subroutine View Load Sample Program Help
Editor Assembler
8085 Assembly Language Editor
Assembler Disassembler
Registers Memory Devices
Memory Editor
Autocorrect
Assemble
Created by : Jubin Mitra

```
// BCD TO HEX CONVERSION
// Manually strore MSD in the memory location C050
// Manually store LSD in the memory location C051
// Result is stored in C052
# BEGIN 0000H
                LXI H,C050
                MOV A,M
                ADD A
                MOV B,A
                ADD A
                ADD A
                ADD B
                INX H
                ADD M
                INX H
                ADD M
                INX H
                MOV M,A
                HLT
# ORG C050
# DB 02,09
```

Memory Range: 0000 ---- FFFF

| Memory Address | Value |
|---|---|
| 0000 | 21 |
| 0001 | 50 |
| 0002 | C0 |
| 0003 | 7E |
| 0004 | 87 |
| 0005 | 47 |
| 0006 | 87 |
| 0007 | 87 |
| 0008 | 80 |
| 0009 | 23 |
| 000A | 86 |
| 000B | 23 |
| 000C | 86 |
| 000D | 23 |
| 000E | 77 |
| 000F | 76 |
| C050 | 02 |
| C051 | 09 |
| C053 | 1D |

○ Show entire content
● Show only loaded memory location
○ Store directly to specified memory location

# 10.WRITE A 8085 PROGRAM TO CONVERT A BCD NUMBER INTO HEXADECIMAL NUMBER

**AIM :** To convert given Hexa decimal number into its equivalent BCD number using 8085 instruction set

## ALGORITHM :

1) Initialize memory pointer to 4150 H
2) Get the Hexa decimal number in C - register
3) Perform repeated addition for C number of times
4) Adjust for BCD in each step
5) Store the BCD data in Memory

## PROGRAM :

```
# BEGIN 0000H
          LXI H,C050     Initialize memory pointer
          MVI D,00       Clear D- reg for Most significant Byte
          XRA A          Clear Accumulator
          MOV C,M        Get HEX data

LOOP2:    ADI 01         Count the number one by one
          DAA            Adjust for BCD count
          JNC LOOP1
          INR D

LOOP1:    DCR C
          JNZ LOOP2
          STA C051       Store the Least Significant Byte
          MOV A,D
          STA C052       STA 4152 Store the Most Significant Byte
          HLT

# ORG C050
# DB FFH
```

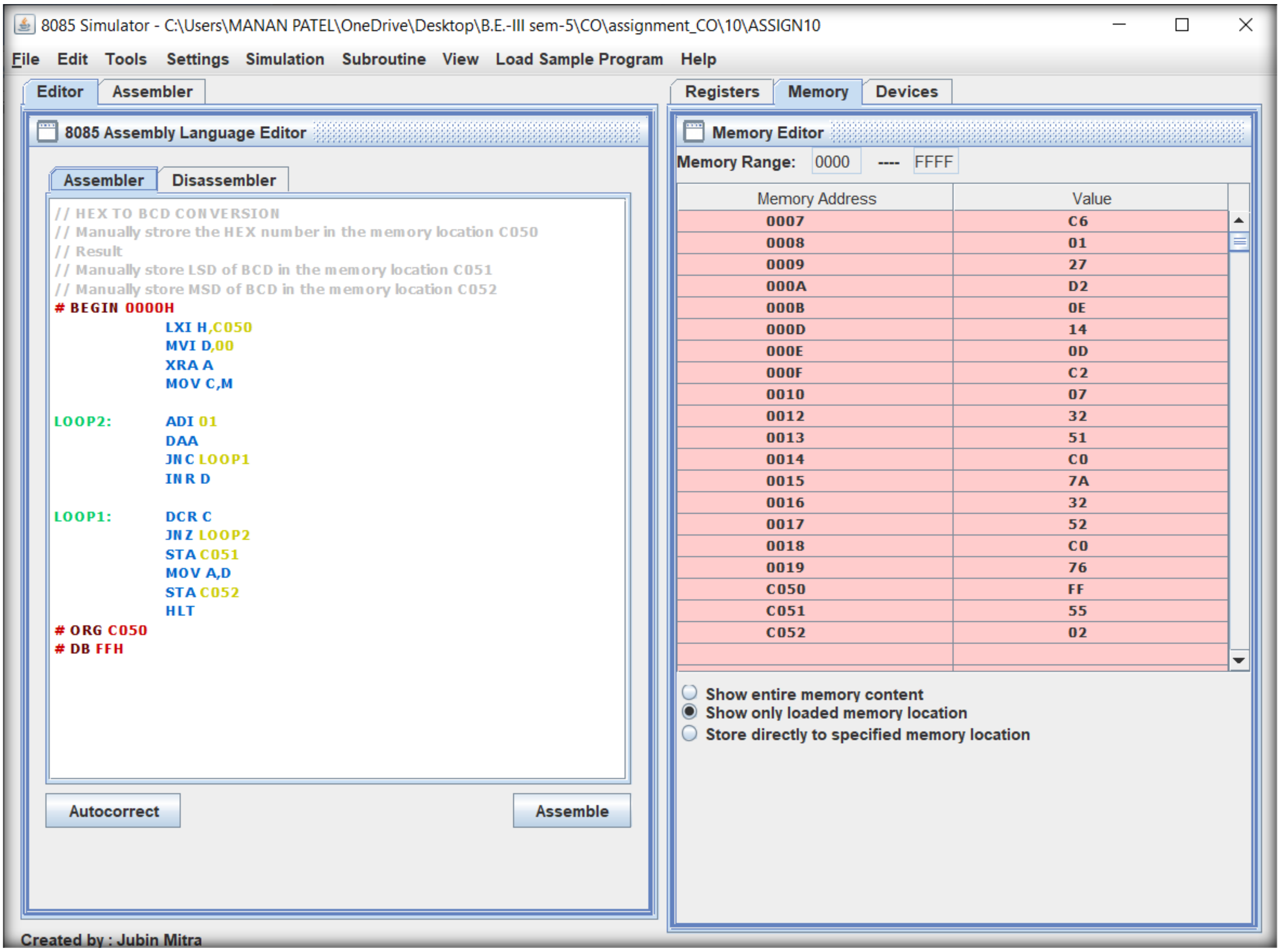
## OBSERVATION :

**Input:** FFH (C050)

**Output:** 55 (C051) -> LSB

02 (C052) -> MSB

**RESULT :** Thus the program to convert HEX data to BCD data was executed.



8085 Simulator - C:\Users\MANAN PATEL\OneDrive\Desktop\B.E.-III sem-5\CO\assignment_CO\10\ASSIGN10
File Edit Tools Settings Simulation Subroutine View Load Sample Program Help
Editor Assembler
8085 Assembly Language Editor
Assembler Disassembler
Registers Memory Devices
Memory Editor
Autocorrect
Assemble
Created by : Jubin Mitra

```
// HEX TO BCD CONVERSION
// Manually strore the HEX number in the memory location C050
// Result
// Manually store LSD of BCD in the memory location C051
// Manually store MSD of BCD in the memory location C052
# BEGIN 0000H
                LXI H,C050
                MVI D,00
                XRA A
                MOV C,M

LOOP2:          ADI 01
                DAA
                JNC LOOP1
                INR D

LOOP1:          DCR C
                JNZ LOOP2
                STA C051
                MOV A,D
                STA C052
                HLT
# ORG C050
# DB FFH
```

Memory Range: 0000 ---- FFFF

| Memory Address | Value |
|---|---|
| 0007 | C6 |
| 0008 | 01 |
| 0009 | 27 |
| 000A | D2 |
| 000B | 0E |
| 000D | 14 |
| 000E | 0D |
| 000F | C2 |
| 0010 | 07 |
| 0012 | 32 |
| 0013 | 51 |
| 0014 | C0 |
| 0015 | 7A |
| 0016 | 32 |
| 0017 | 52 |
| 0018 | C0 |
| 0019 | 76 |
| C050 | FF |
| C051 | 55 |
| C052 | 02 |

○ Show entire memory content
● Show only loaded memory location
○ Store directly to specified memory location

**THANK YOU !!!**