Embedded System Lab

(ELC3930)

Experiment No.: _01_

Object: **Write a program using 8085 simulator to arrange N numbers in (a) Ascending order; (b) Descending order using Bubble sort and Selection sort algorithm.**

G. No: GL3136

S. No: A3EL-02

F. No: 19ELB056

Name: Maha Zakir Khan

Date of performing experiment: 08 | 01 | 2020

Date of report submission: 16 | 01 | 2020

Simulator Used:

8085 Simulator by Jubin Mitra. It helps in get started easily with example codes, and to learn the architecture playfully. This tool is an integrated software environment for teaching microprocessor concepts. The software is shared under opensource GNU license.

Link: [8085 Jubin Simulator]

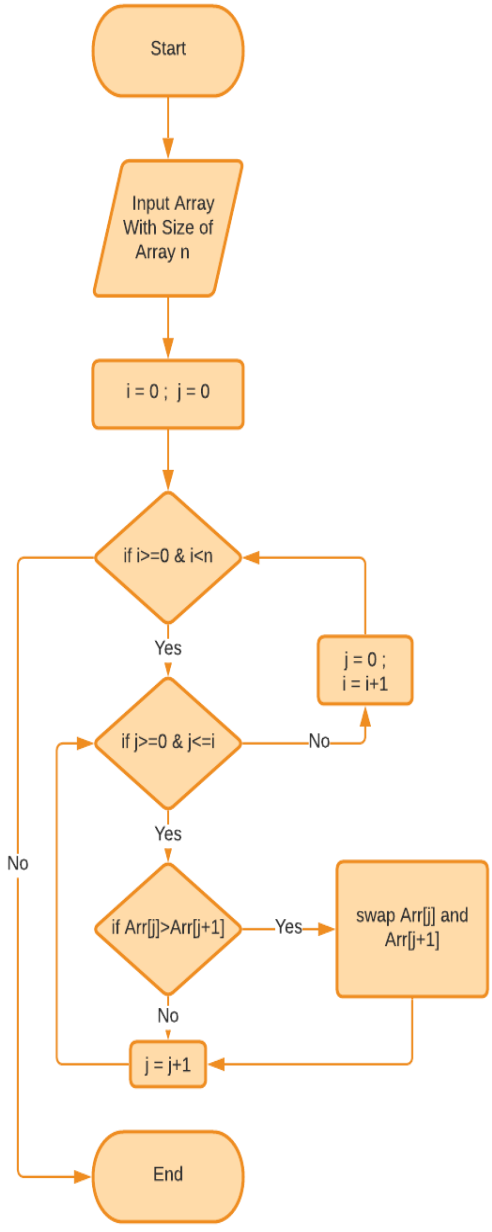
Algorithm:

1) **Bubble Sort –** For an array Arr on N cells, it works by swapping two adjacent cells depending on a particular condition consecutively till the end of the array and the process is repeated for the array again and again for N-1, N-2 …. 0 cells, till the array is sorted.
    a) For Ascending Order, the special condition for swapping is Arr[i]>Arr[i+1]
    b) For Descending Order, the special condition for swapping is Arr[i]<Arr[i+1]
2) **Selection Sort –** This algorithm finds the minimum/maximum of a subarray of the array Arr with elements N and swaps it with the beginning/ending of the subarray. This process is repeated for next N-1, N-2….0 cells consecutive subarrays till the array is sorted
    a) For Ascending Order, we find the minimum of a subarray and swap the minimum element with the beginning of the subarray
    b) For Descending Order, we find the maximum of a subarray and swap the maximum element with the beginning of the subarray

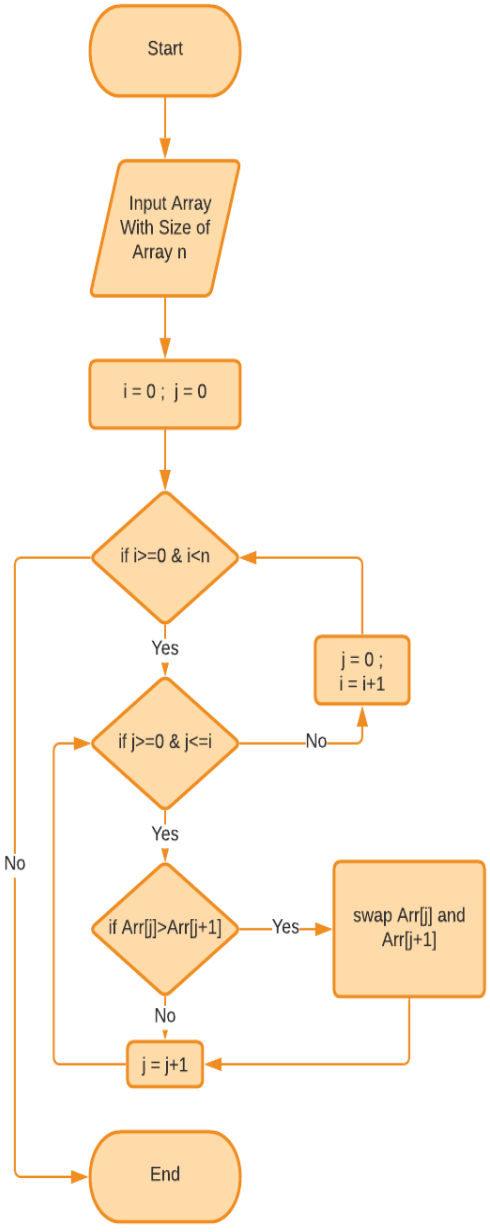
Flow Chart:

1) **Bubble Sort:**

   a) **Ascending Order -**



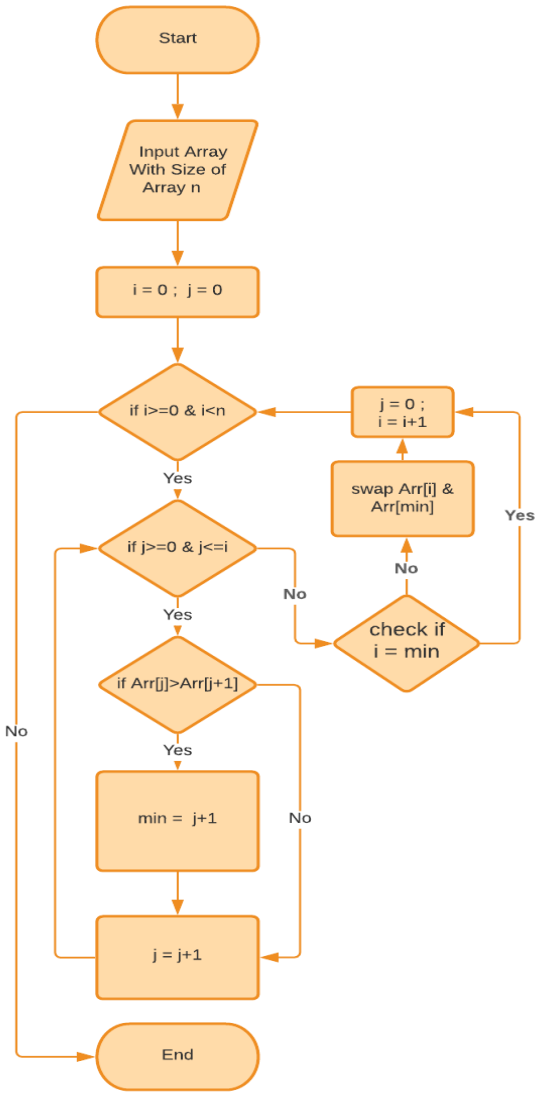
   b) **Descending Order -**



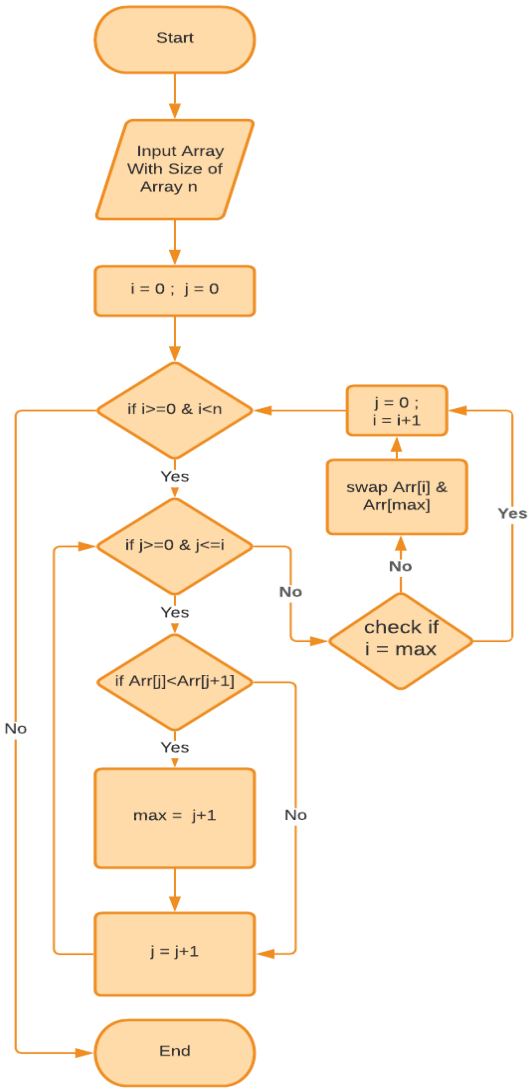
* if Arr[j]<Arr[j+1]

2) Selection Sort:

a) Ascending Order -



b) Descending Order -



Program:

1) Bubble Sort:

a) Ascending Order -

```
#ORG 5050H
INITIALIZE:  // Loading outer loop counter
LXI H, 5000H
MOV D,M
DCR D

START:
LXI H,5000H
MOV C,D // Loading inner loop counter
        INX H
        DCR D
        JZ STOP
        COMPARE: // Compares adjacent cells and sends to SWAP on no carry otherwise jumps back to START
        MOV A,M
        INX H
        CMP M
        JNC SWAP
        DCR C
        JZ START
        JC COMPARE
        SWAP: // Exchange two adjacent memory cells
        MOV B,M
        MOV M,A
        DCX H
        MOV M,B
        INX H
        DCR C
        JNZ COMPARE
        JZ START
        STOP: // Terminate program
        HLT
        #ORG 5000H
        #DB 08H, 08H,07H,03H,05H,02H,11H,07H,04H // Load data into memory
```

b) Descending Order -

```
1. #ORG 5050H
2. INITIALIZE:  // Loading outer loop counter
3. LXI H, 5000H
```

```
MOV D,M
DCR D

START:
LXI H,5000H
MOV C,D // Loading inner loop counter
INX H
DCR D
JZ STOP
COMPARE: // Compares adjacent cells and sends to SWAP on no carry otherwise jumps back to START
MOV A,M
INX H
CMP M
JC SWAP
DCR C
JZ START
JNC COMPARE
SWAP: // Exchange two adjacent memory cells
MOV B,M
MOV M,A
DCX H
MOV M,B
INX H
DCR C
JNZ COMPARE
JZ START
STOP: // Terminate program
HLT
#ORG 5000H
#DB 08H, 08H,07H,03H,05H,02H,11H,07H,04H // Load data into memory
```

3) Selection Sort:

a) Ascending Order -

```
# ORG 6000H

        // Loading outer loop counter B with size of data
INITIALIZE:
            LXI H,5000
  MOV B,M
  MOV C,B
  INX H

  // Loading accumulator with memory at HL address
START:
            MOV A,M
  PUSH H
  INX H
  DCR B
  JZ STOP
  MOV C,B       // Loading inner loop counter with B
  CALL MIN
  POP H
  CMP M
  JC SWAP
            INX H
  JMP START

  // Exchange current memory cell with min value cell
SWAP:   MOV A,M
  XCHG
  MOV C,M
  MOV M,A
  XCHG
  MOV M,C
  INX H
  JMP START

  // Finds min of a subarray and stores the address in DE reg pair
MIN:    CMP M
  JC CONTINUE
  PUSH H
  POP D
  MOV A,M
  INX H
  DCR C
  RZ
  JMP MIN
```

```
45.
46.         CONTINUE:       INX H
47.           DCR C
48.           RZ
49.           JMP MIN
50.
51.         // Terminate the program
52.         STOP:     HLT
53.
54.         // Load data into memory
55.         # ORG 5000H
56.         # DB 05H,11H,15H,14H,10H,04H
```

b) Descending Order -

```
1. # ORG 6000H
2.
3.          // Loading outer loop counter B with size of data
4.          INITIALIZE:
5.                      LXI H,5000
6.            MOV B,M
7.            MOV C,B
8.            INX H


9.          // Loading accumulator with memory at HL address
10.           START:
11.                     MOV A,M
12.             PUSH H
13.             INX H
14.             DCR B
15.             JZ STOP
16.             MOV C,B       // Loading inner loop counter with B
17.             CALL MIN
18.             POP H
19.             CMP M
20.             JNC SWAP
21.                     INX H
22.             JMP START


23.         // Exchange current memory cell with max value cell
24.         SWAP:     MOV A,M
25.           XCHG
```

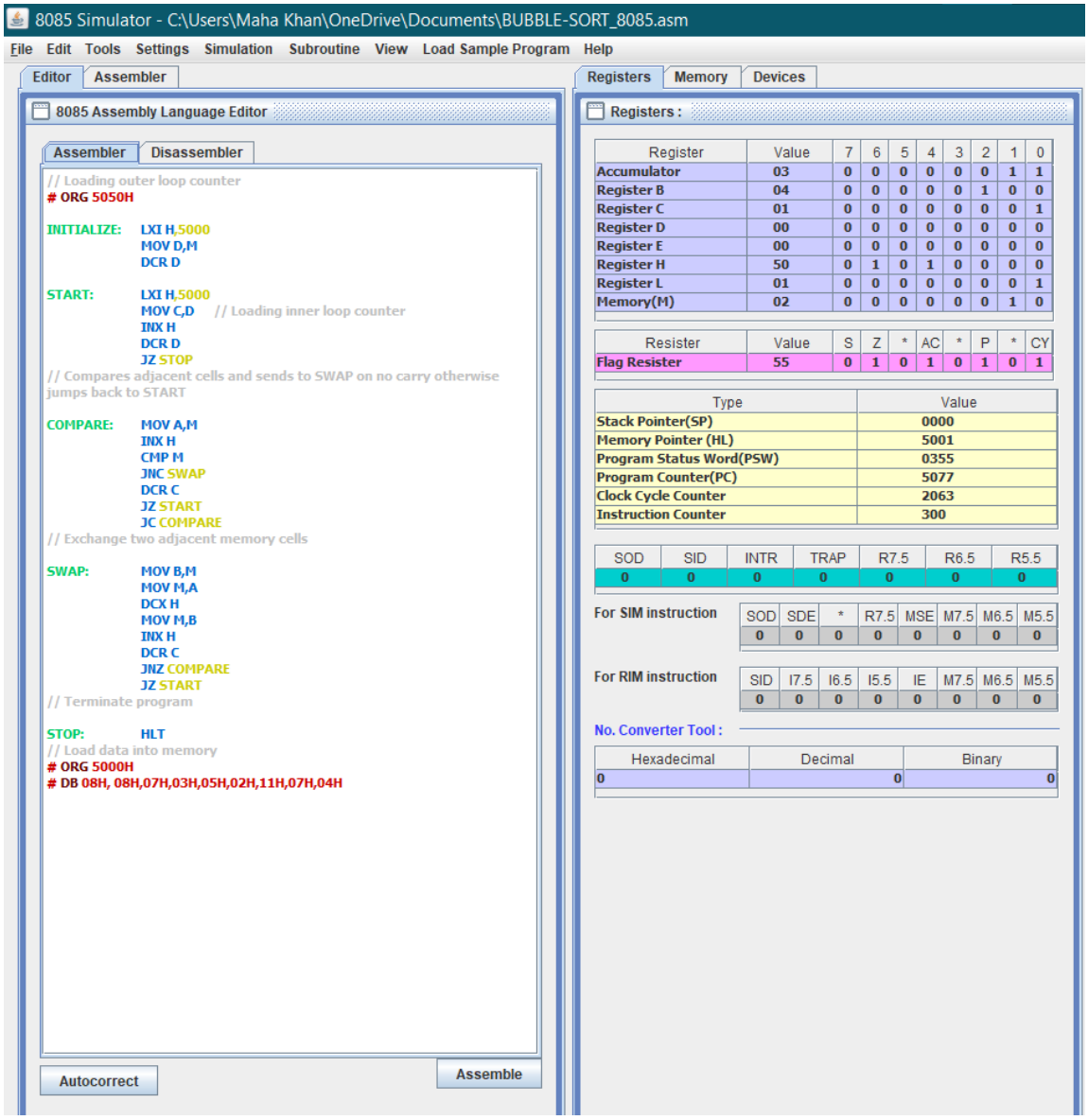
```
26.         MOV C,M
27.         MOV M,A
28.         XCHG
29.         MOV M,C
30.         INX H
31.         JMP START


32.     // Finds max of a subarray and stores the address in DE reg pair
33.     MIN:    CMP M
34.       JNC CONTINUE
35.       PUSH H
36.       POP D
37.       MOV A,M
38.       INX H
39.       DCR C
40.       RZ
41.       JMP MIN
42.
43.     CONTINUE:
44.             INX H
45.       DCR C
46.       RZ
47.       JMP MIN
48.
49.     // Terminate the program
50.     STOP:   HLT
51.
52.     // Load data into memory
53.     # ORG 5000H
54.     # DB 05H,11H,15H,14H,10H,04H
```

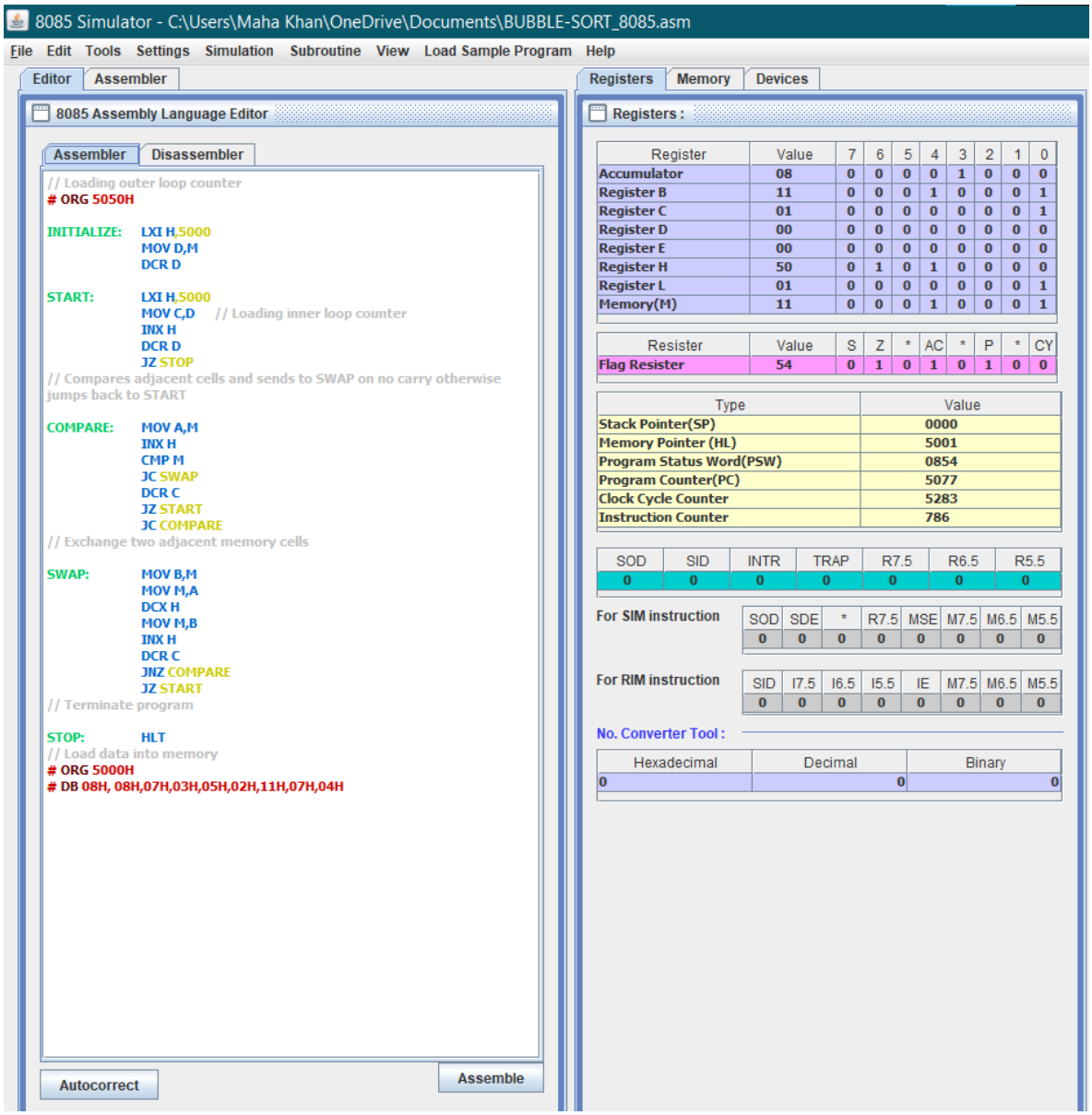
Screen-grab of Simulator:

1) Bubble Sort:

a) Ascending Order -



b) Descending Order -



Selection Sort:

c) Ascending Order

8085 Simulator - C:\Users\Maha Khan\OneDrive\Documents\SELECTION-SORT_8085.asm

File Edit Tools Settings Simulation Subroutine View Load Sample Program Help

Editor | Assembler

8085 Assembly Language Editor

Assembler | Disassembler

```
// Loading outer loop counter B with size of data
# ORG 6000H

INITIALIZE:   LXI H,5000
              MOV B,M
              MOV C,B
              INX H
// Loading accumulator with memory at HL address

START:        MOV A,M
              PUSH H
              INX H
              DCR B
              JZ STOP
              MOV C,B     // Loading inner loop counter with B
              CALL MIN
              POP H
              CMP M
              JC SWAP
              INX H
              JMP START
// Excahnge current memory cell with min value cell

SWAP:         MOV A,M
              XCHG
              MOV C,M
              MOV M,A
              XCHG
              MOV M,C
              INX H
              JMP START
// Finds min of a subarray and stores the address in DE reg pair

MIN:          CMP M
              JC CONTINUE
              PUSH H
              POP D
              MOV A,M
              INX H
              DCR C
              RZ
              JMP MIN

CONTINUE:     INX H
              DCR C
              RZ
              JMP MIN
// Terminate the program

STOP:         HLT
// Load data into memory
# ORG 5000H
# DB 05H,11H,05H,14H,10H,04H
```

Autocorrect | Assemble

Registers | Memory | Devices

Registers :

Register	Value	7	6	5	4	3	2	1	0
Accumulator	00	0	0	0	0	0	0	0	0
Register B	00	0	0	0	0	0	0	0	0
Register C	00	0	0	0	0	0	0	0	0
Register D	00	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	00	0	0	0	0	0	0	0	0
Register L	00	0	0	0	0	0	0	0	0
Memory(M)	00	0	0	0	0	0	0	0	0

Resister	Value	S	Z	*	AC	*	P	*	CY
Flag Resister	00	0	0	0	0	0	0	0	0

Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	0000
Program Status Word(PSW)	0000
Program Counter(PC)	6000
Clock Cycle Counter	0
Instruction Counter	0

SOD	SID	INTR	TRAP	R7.5	R6.5	R5.5
0	0	0	0	0	0	0

For SIM instruction

SOD	SDE	*	R7.5	MSE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

For RIM instruction

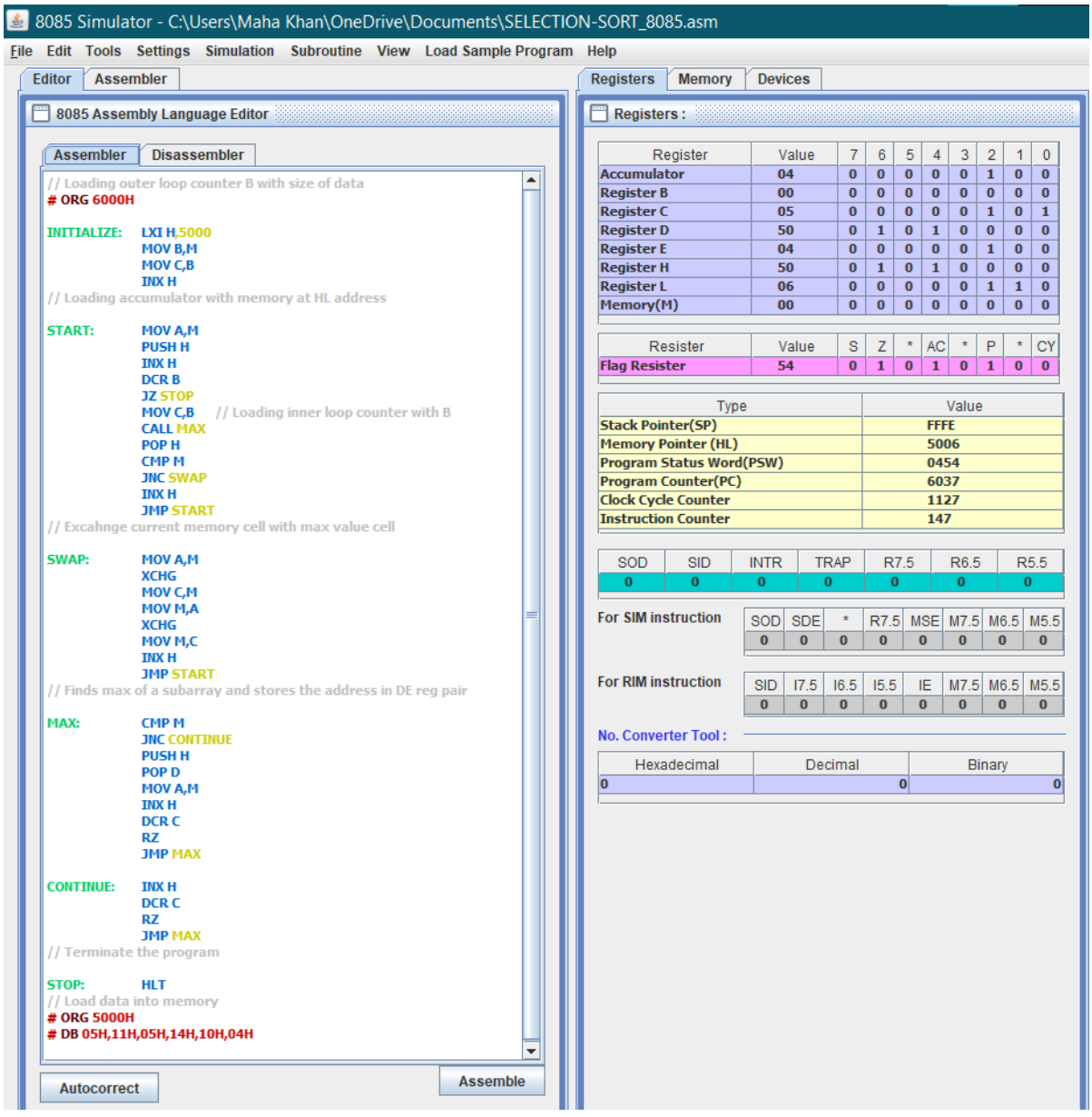
SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5
0	0	0	0	0	0	0	0

No. Converter Tool :

Hexadecimal	Decimal	Binary
0	0	0

-

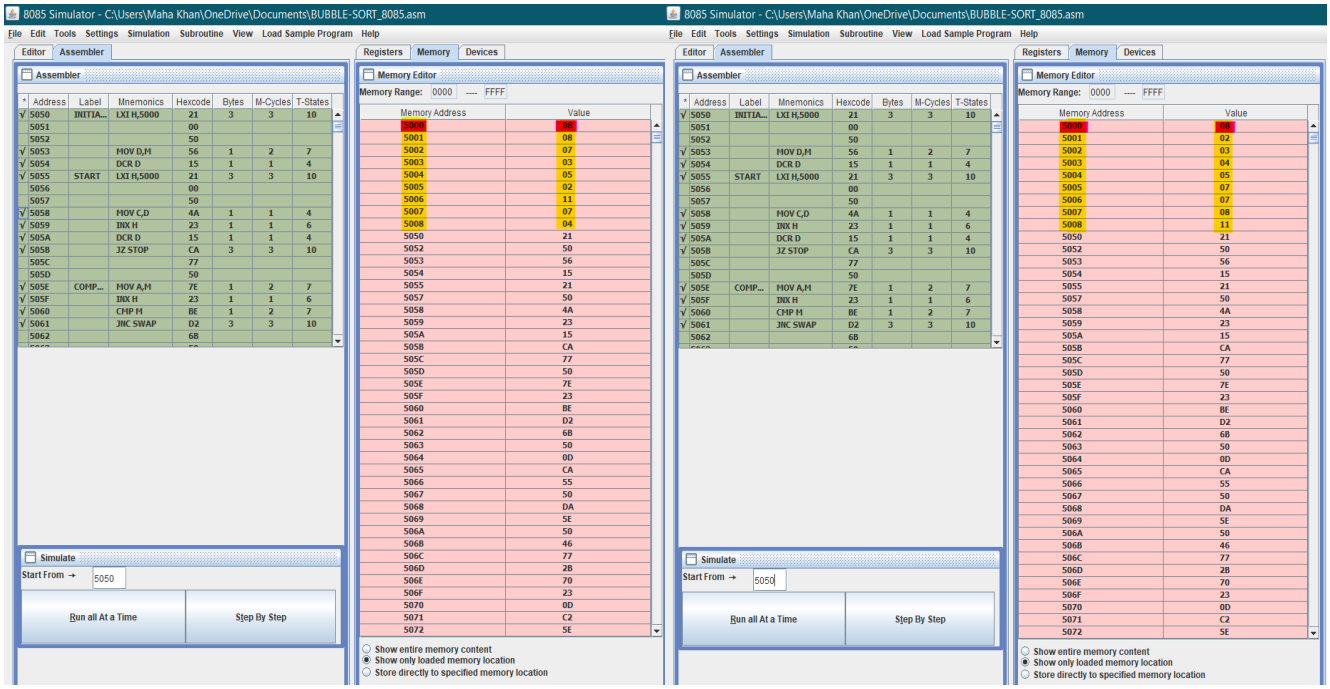
d) Descending Order -



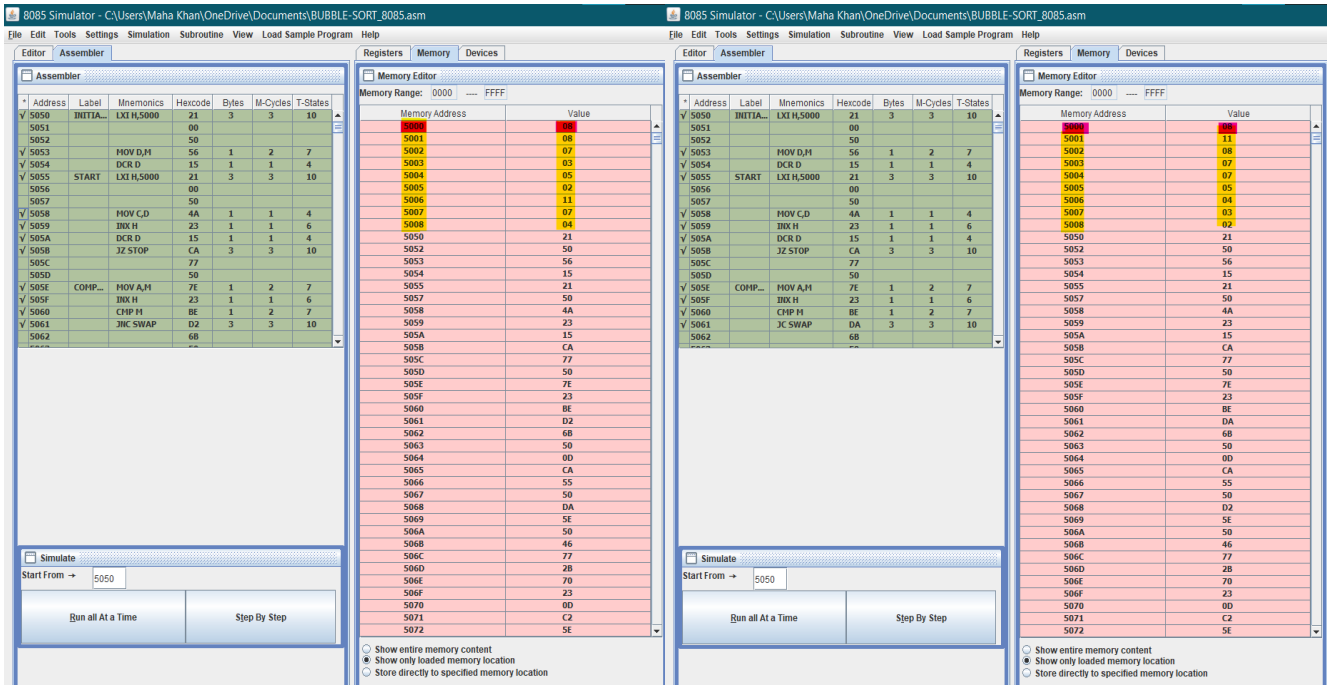
Result:

1) Bubble Sort:

a) Ascending Order -

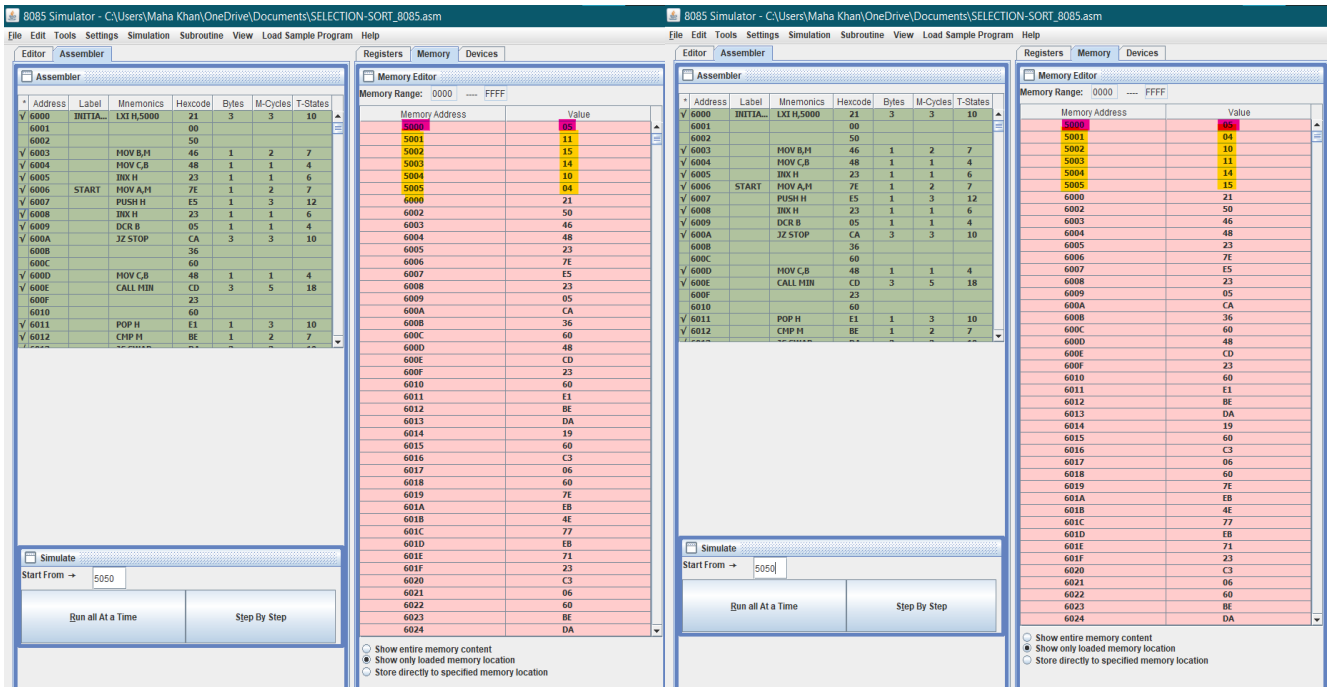


b) Descending Order -

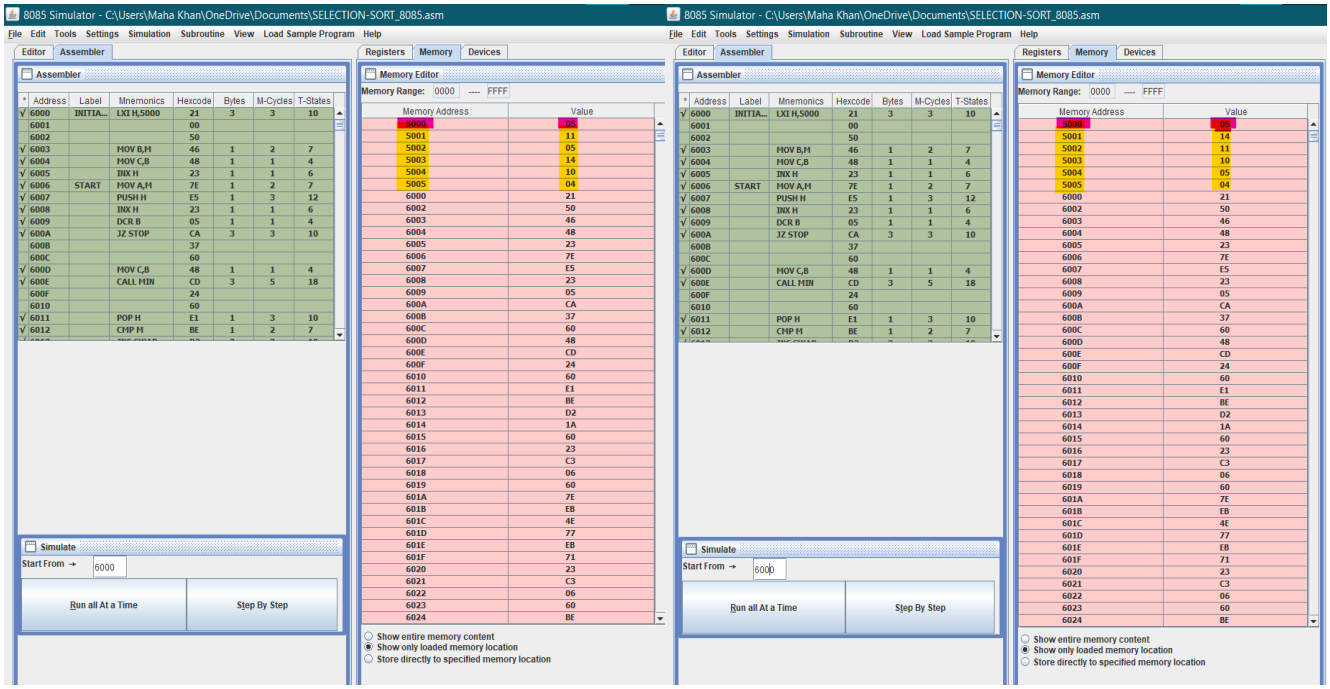


2) Selection Sort:

a) Ascending Order -



b) Descending Order -



Discussion:

In Bubble sorting since there are two nested loops that need to be maintained; the worst-case complexity comes out to be O(n^2). The best complexity of a bubble sort can be O(n). O(n) is only possible if the array is sorted. As we can observe, it is one of the simplest but most inefficient sorting algorithm since we need to create temporary variable for each adjacent swapping of cells. We can optimize the algorithm using a swap flag variable that exits the loop once swapping is done.

Selection sort algorithm is efficient as compared to bubble sort. It uses lesser swaps and has an O(n^2) time complexity; however, this makes it which makes it inefficient on large lists, and generally performs worse than the similar insertion sort.

I used a stack for storing minimum/maximum data address as well as the memory address of the current cell before jumping into the inner loop in 8085 assembly code