

# 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](#)

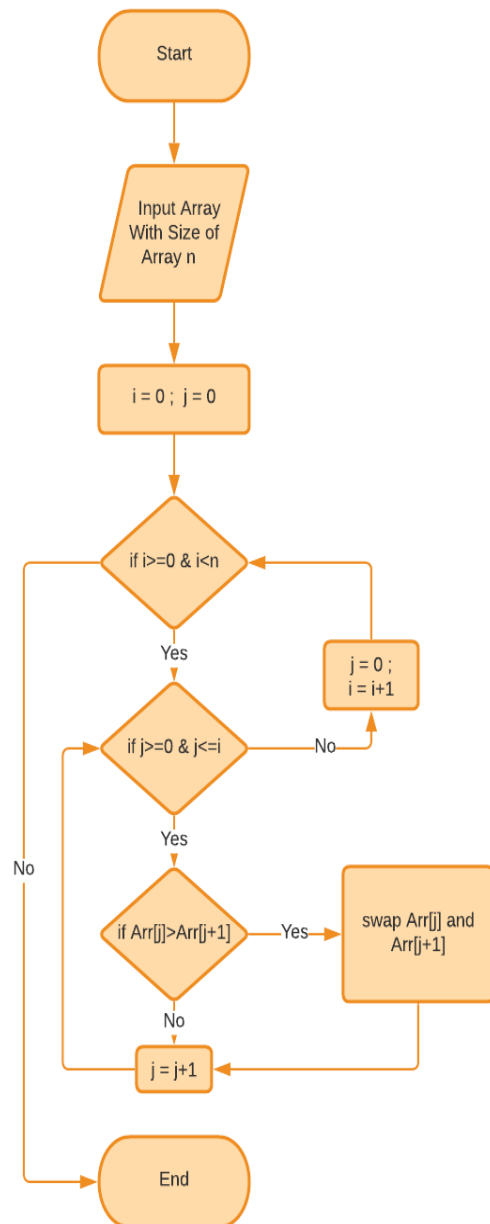
## Algorithm:

- 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

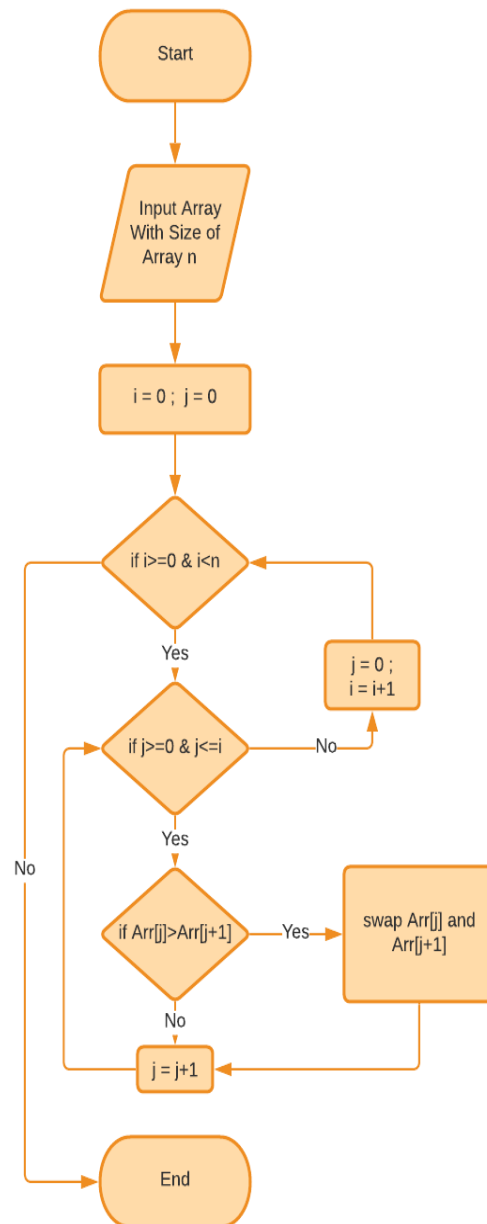
# Flow Chart:

## 1) Bubble Sort:

### a) Ascending Order -



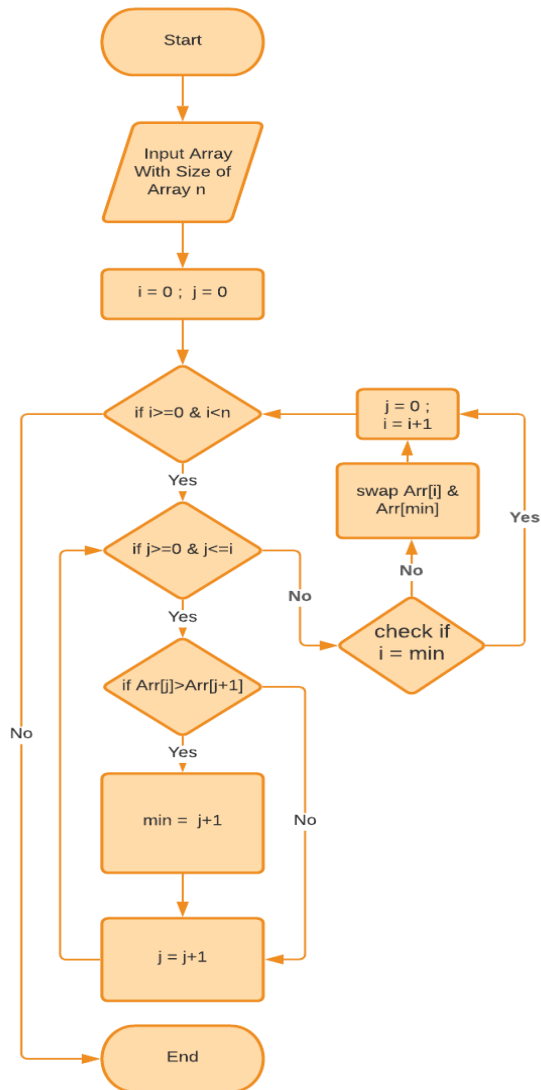
### b) Descending Order -



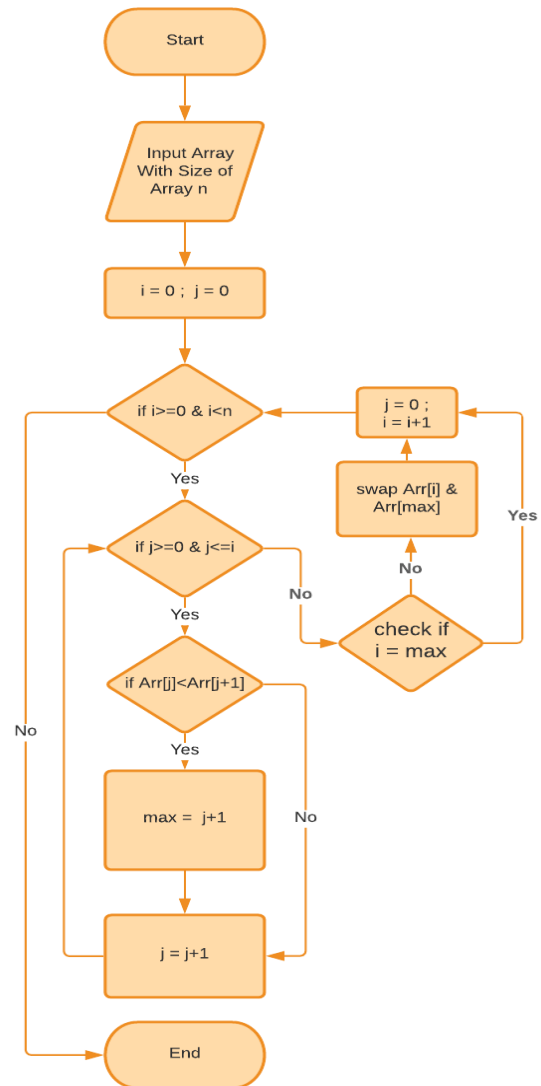
\* if  $\text{Arr}[j] < \text{Arr}[j + 1]$

## 2) Selection Sort:

### a) Ascending Order -



### b) Descending Order -



# Program:

## 1) Bubble Sort:

### a) Ascending Order -

```
1. #ORG 5050H
2. INITIALIZE: // Loading outer loop counter
3. LXI H, 5000H
4. MOV D,M
5. DCR D
6.
7. START:
8. LXI H,5000H
9. MOV C,D // Loading inner loop counter
10.     INX H
11.     DCR D
12.     JZ STOP
13.     COMPARE: // Compares adjacent cells and sends to SWAP
                on no carry otherwise jumps back to START
14.     MOV A,M
15.     INX H
16.     CMP M
17.     JNC SWAP
18.     DCR C
19.     JZ START
20.     JC COMPARE
21.     SWAP: // Exchange two adjacent memory cells
22.     MOV B,M
23.     MOV M,A
24.     DCX H
25.     MOV M,B
26.     INX H
27.     DCR C
28.     JNZ COMPARE
29.     JZ START
30.     STOP: // Terminate program
31.     HLT
32.     #ORG 5000H
33.     #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
```

```
4.  MOV D,M
5.  DCR D
6.
7.  START:
8.  LXI H,5000H
9.  MOV C,D // Loading inner loop counter
10. INX H
11. DCR D
12. JZ STOP
13. COMPARE: // Compares adjacent cells and sends to SWAP on no
    carry otherwise jumps back to START
14. MOV A,M
15. INX H
16. CMP M
17. JC SWAP
18. DCR C
19. JZ START
20. JNC COMPARE
21. SWAP: // Exchange two adjacent memory cells
22. MOV B,M
23. MOV M,A
24. DCX H
25. MOV M,B
26. INX H
27. DCR C
28. JNZ COMPARE
29. JZ START
30. STOP: // Terminate program
31. HLT
32. #ORG 5000H
33. #DB 08H, 08H,07H,03H,05H,02H,11H,07H,04H // Load data into
    memory
```

### 3) Selection Sort:

#### a) Ascending 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.
10.     // Loading accumulator with memory at HL address
11.     START:
12.         MOV A,M
13.         PUSH H
14.         INX H
15.         DCR B
16.         JZ STOP
17.         MOV C,B      // Loading inner loop counter with B
18.         CALL MIN
19.         POP H
20.         CMP M
21.         JC SWAP
22.         INX H
23.         JMP START
24.
25.     // Exchange current memory cell with min value cell
26.     SWAP:  MOV A,M
27.         XCHG
28.         MOV C,M
29.         MOV M,A
30.         XCHG
31.         MOV M,C
32.         INX H
33.         JMP START
34.
35.     // Finds min of a subarray and stores the address in DE reg pair
36.     MIN:  CMP M
37.         JC CONTINUE
38.         PUSH H
39.         POP D
40.         MOV A,M
41.         INX H
42.         DCR C
43.         RZ
44.         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 -

8085 Simulator - C:\Users\Maha Khan\OneDrive\Documents\BUBBLE-SORT\_8085.asm

File Edit Tools Settings Simulation Subroutine View Load Sample Program Help

Editor Assembler Registers Memory Devices

8085 Assembly Language Editor

Assembler Disassembler

```
// Loading outer loop counter
# ORG 5050H

INITIALIZE: LXI H,5000
            MOV D,M
            DCR D

START:      LXI H,5000
            MOV C,D // Loading inner loop counter
            INX H
            DCR D
            JZ STOP

// Compares adjacent cells and sends to SWAP on no carry otherwise
// jumps back to START

COMPARE:    MOV A,M
            INX H
            CMP M
            JNC SWAP
            DCR C
            JZ START
            JC COMPARE

// Exchange two adjacent memory cells

SWAP:       MOV B,M
            MOV M,A
            DCX H
            MOV M,B
            INX H
            DCR C
            JNZ COMPARE
            JZ START

// Terminate program

STOP:       HLT

// Load data into memory
# ORG 5000H
# DB 08H, 08H,07H,03H,05H,02H,11H,07H,04H
```

Autocorrect Assemble

Registers :

Register	Value	7	6	5	4	3	2	1	0
Accumulator	03	0	0	0	0	0	0	1	1
Register B	04	0	0	0	0	0	1	0	0
Register C	01	0	0	0	0	0	0	0	1
Register D	00	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	50	0	1	0	1	0	0	0	0
Register L	01	0	0	0	0	0	0	0	1
Memory(M)	02	0	0	0	0	0	0	1	0

Register	Value	S	Z	*	AC	*	P	*	CY
Flag Register	55	0	1	0	1	0	1	0	1

Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	5001
Program Status Word(PSW)	0355
Program Counter(PC)	5077
Clock Cycle Counter	2063
Instruction Counter	300

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

## b) Descending Order -

8085 Simulator - C:\Users\Maha Khan\OneDrive\Documents\BUBBLE-SORT\_8085.asm

File Edit Tools Settings Simulation Subroutine View Load Sample Program Help

Editor Assembler Registers Memory Devices

8085 Assembly Language Editor

Assembler Disassembler

```
// Loading outer loop counter
# ORG 5050H

INITIALIZE: LXI H,5000
            MOV D,M
            DCR D

START:      LXI H,5000
            MOV C,D // Loading inner loop counter
            INX H
            DCR D
            JZ STOP

// Compares adjacent cells and sends to SWAP on no carry otherwise
jumps back to START

COMPARE:    MOV A,M
            INX H
            CMP M
            JC SWAP
            DCR C
            JZ START
            JC COMPARE

// Exchange two adjacent memory cells

SWAP:       MOV B,M
            MOV M,A
            DCX H
            MOV M,B
            INX H
            DCR C
            JNZ COMPARE
            JZ START

// Terminate program

STOP:       HLT

// Load data into memory
# ORG 5000H
# DB 08H, 08H,07H,03H,05H,02H,11H,07H,04H
```

Autocorrect Assemble

Registers :

Register	Value	7	6	5	4	3	2	1	0
Accumulator	08	0	0	0	0	1	0	0	0
Register B	11	0	0	0	1	0	0	0	1
Register C	01	0	0	0	0	0	0	0	1
Register D	00	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	50	0	1	0	1	0	0	0	0
Register L	01	0	0	0	0	0	0	0	1
Memory(M)	11	0	0	0	1	0	0	0	1

Register	Value	S	Z	*	AC	*	P	*	CY
Flag Register	54	0	1	0	1	0	1	0	0

Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	5001
Program Status Word(PSW)	0854
Program Counter(PC)	5077
Clock Cycle Counter	5283
Instruction Counter	786

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

## 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

// 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

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

Register	Value	S	Z	*	AC	*	P	*	CY
Flag Register	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 -

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 MAX
            POP H
            CMP M
            JNC SWAP
            INX H
            JMP START

// Exchange current memory cell with max value cell

SWAP:       MOV A,M
            XCHG
            MOV C,M
            MOV M,A
            XCHG
            MOV M,C
            INX H
            JMP START

// Finds max of a subarray and stores the address in DE reg pair

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

CONTINUE:   INX H
            DCR C
            RZ
            JMP MAX

// 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	04	0	0	0	0	0	1	0	0
Register B	00	0	0	0	0	0	0	0	0
Register C	05	0	0	0	0	0	1	0	1
Register D	50	0	1	0	1	0	0	0	0
Register E	04	0	0	0	0	0	1	0	0
Register H	50	0	1	0	1	0	0	0	0
Register L	06	0	0	0	0	0	1	1	0
Memory(M)	00	0	0	0	0	0	0	0	0

Register	Value	S	Z	*	AC	*	P	*	CY
Flag Register	54	0	1	0	1	0	1	0	0

Type	Value
Stack Pointer(SP)	FFFE
Memory Pointer (HL)	5006
Program Status Word(PSW)	0454
Program Counter(PC)	6037
Clock Cycle Counter	1127
Instruction Counter	147

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

# Result:

## 1) Bubble Sort:

### a) Ascending Order -

The screenshot displays the 8085 Simulator interface for the Bubble Sort program in Ascending Order. The Assembler window shows the following code:

Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
5050	INITIA...	LXI H,5000	21	3	3	10
5051			00			
5052			50			
5053		MOV D,H	56	1	2	7
5054		DCR D	15	1	1	4
5055	START	LXI H,5000	21	3	3	10
5056			00			
5057			50			
5058		MOV C,D	4A	1	1	4
5059		INX H	23	1	1	6
505A		DCR D	15	1	1	4
505B		JZ STOP	CA	3	3	10
505C			77			
505D			50			
505E	COMP...	MOV A,H	7E	1	2	7
505F		INX H	23	1	1	6
5060		CHP H	BE	1	2	7
5061		JNC SWAP	D2	3	3	10
5062			68			

The Memory Editor window shows the memory contents after execution. The memory range is 0000 to FFFF. The memory address 5000 contains the value 00, and the memory address 5001 contains the value 08. The memory address 5002 contains the value 07, and the memory address 5003 contains the value 05. The memory address 5004 contains the value 05, and the memory address 5005 contains the value 02. The memory address 5006 contains the value 11, and the memory address 5007 contains the value 07. The memory address 5008 contains the value 04, and the memory address 5009 contains the value 21. The memory address 5050 contains the value 50, and the memory address 5051 contains the value 15. The memory address 5052 contains the value 50, and the memory address 5053 contains the value 56. The memory address 5054 contains the value 15, and the memory address 5055 contains the value 21. The memory address 5056 contains the value 50, and the memory address 5057 contains the value 50. The memory address 5058 contains the value 4A, and the memory address 5059 contains the value 23. The memory address 505A contains the value 15, and the memory address 505B contains the value CA. The memory address 505C contains the value 77, and the memory address 505D contains the value 50. The memory address 505E contains the value 7E, and the memory address 505F contains the value 23. The memory address 5060 contains the value BE, and the memory address 5061 contains the value D2. The memory address 5062 contains the value 68, and the memory address 5063 contains the value 50. The memory address 5064 contains the value 00, and the memory address 5065 contains the value CA. The memory address 5066 contains the value 55, and the memory address 5067 contains the value 50. The memory address 5068 contains the value DA, and the memory address 5069 contains the value SE. The memory address 506A contains the value 50, and the memory address 506B contains the value 46. The memory address 506C contains the value 77, and the memory address 506D contains the value 2B. The memory address 506E contains the value 70, and the memory address 506F contains the value 23. The memory address 5070 contains the value 00, and the memory address 5071 contains the value C2. The memory address 5072 contains the value SE.

### b) Descending Order -

The screenshot displays the 8085 Simulator interface for the Bubble Sort program in Descending Order. The Assembler window shows the following code:

Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
5050	INITIA...	LXI H,5000	21	3	3	10
5051			00			
5052			50			
5053		MOV D,H	56	1	2	7
5054		DCR D	15	1	1	4
5055	START	LXI H,5000	21	3	3	10
5056			00			
5057			50			
5058		MOV C,D	4A	1	1	4
5059		INX H	23	1	1	6
505A		DCR D	15	1	1	4
505B		JZ STOP	CA	3	3	10
505C			77			
505D			50			
505E	COMP...	MOV A,H	7E	1	2	7
505F		INX H	23	1	1	6
5060		CHP H	BE	1	2	7
5061		JNC SWAP	DA	3	3	10
5062			68			

The Memory Editor window shows the memory contents after execution. The memory range is 0000 to FFFF. The memory address 5000 contains the value 00, and the memory address 5001 contains the value 08. The memory address 5002 contains the value 07, and the memory address 5003 contains the value 05. The memory address 5004 contains the value 05, and the memory address 5005 contains the value 02. The memory address 5006 contains the value 11, and the memory address 5007 contains the value 07. The memory address 5008 contains the value 04, and the memory address 5009 contains the value 21. The memory address 5050 contains the value 50, and the memory address 5051 contains the value 15. The memory address 5052 contains the value 50, and the memory address 5053 contains the value 56. The memory address 5054 contains the value 15, and the memory address 5055 contains the value 21. The memory address 5056 contains the value 50, and the memory address 5057 contains the value 50. The memory address 5058 contains the value 4A, and the memory address 5059 contains the value 23. The memory address 505A contains the value 15, and the memory address 505B contains the value CA. The memory address 505C contains the value 77, and the memory address 505D contains the value 50. The memory address 505E contains the value 7E, and the memory address 505F contains the value 23. The memory address 5060 contains the value BE, and the memory address 5061 contains the value DA. The memory address 5062 contains the value 68, and the memory address 5063 contains the value 50. The memory address 5064 contains the value 00, and the memory address 5065 contains the value CA. The memory address 5066 contains the value 55, and the memory address 5067 contains the value 50. The memory address 5068 contains the value DA, and the memory address 5069 contains the value SE. The memory address 506A contains the value 50, and the memory address 506B contains the value 46. The memory address 506C contains the value 77, and the memory address 506D contains the value 2B. The memory address 506E contains the value 70, and the memory address 506F contains the value 23. The memory address 5070 contains the value 00, and the memory address 5071 contains the value C2. The memory address 5072 contains the value SE.

a) **Ascending Order -**

8085 Simulator - C:\Users\Nisha Khan\OneDrive\Documents\SELECTION\_SORT\_8085.asm

FileEditToolsSettingsSimulationSubroutineViewLoad Sample ProgramHelp

EditorAssembler

Assembler

* Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
✓ 6000	INITIAL...	LXI H,5000	21	3	3	10
6001		00				
6002		50				
✓ 6003		MOV B,H	46	1	2	7
✓ 6004		MOV C,B	48	1	1	4
✓ 6005		INX H	23	1	1	6
✓ 6006	START	MOV A,H	7E	1	2	7
✓ 6007		PUSH H	E5	1	3	12
✓ 6008		INX H	23	1	1	6
✓ 6009		DCR B	05	1	1	4
✓ 600A		JZ STOP	CA	3	3	10
600B			36			
600C			60			
✓ 600D		MOV C,B	48	1	1	4
✓ 600E		CALL MIN	CD	3	5	18
600F			23			
6010			60			
✓ 6011		POP H	E1	1	3	10
✓ 6012		CHP H	BE	1	2	7

Simulate

Start From → 5050

Run all At a Time

Step By Step

RegistersMemoryDevices

Memory Editor

Memory Range: 0000 --- FFFF

Memory Address	Value
5000	05
5001	11
5002	15
5003	14
5004	10
5005	04
6000	21
6001	50
6002	46
6003	48
6004	23
6005	28
6006	7E
6007	E5
6008	23
6009	05
600A	CA
600B	36
600C	60
600D	48
600E	CD
600F	23
6010	60
6011	E1
6012	BE
6013	DA
6014	19
6015	60
6016	C3
6017	06
6018	60
6019	7E
601A	EB
601B	4E
601C	77
601D	EB
601E	71
601F	23
6020	C3
6021	06
6022	60
6023	BE
6024	DA

☐ Show entire memory content  
☒ Show only loaded memory location  
☐ Store directly to specified memory location

8085 Simulator - C:\Users\Nisha Khan\OneDrive\Documents\SELECTION\_SORT\_8085.asm

FileEditToolsSettingsSimulationSubroutineViewLoad Sample ProgramHelp

EditorAssembler

Assembler

* Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
✓ 6000	INITIAL...	LXI H,5000	21	3	3	10
6001		00				
6002		50				
✓ 6003		MOV B,H	46	1	2	7
✓ 6004		MOV C,B	48	1	1	4
✓ 6005		INX H	23	1	1	6
✓ 6006	START	MOV A,H	7E	1	2	7
✓ 6007		PUSH H	E5	1	3	12
✓ 6008		INX H	23	1	1	6
✓ 6009		DCR B	05	1	1	4
✓ 600A		JZ STOP	CA	3	3	10
600B			36			
600C			60			
✓ 600D		MOV C,B	48	1	1	4
✓ 600E		CALL MIN	CD	3	5	18
600F			23			
6010			60			
✓ 6011		POP H	E1	1	3	10
✓ 6012		CHP H	BE	1	2	7

Simulate

Start From → 5050

Run all At a Time

Step By Step

RegistersMemoryDevices

Memory Editor

Memory Range: 0000 --- FFFF

Memory Address	Value
5000	05
5001	04
5002	10
5003	11
5004	14
5005	15
6000	21
6001	50
6002	46
6003	48
6004	23
6005	28
6006	7E
6007	E5
6008	23
6009	05
600A	CA
600B	36
600C	60
600D	48
600E	CD
600F	23
6010	60
6011	E1
6012	BE
6013	DA
6014	19
6015	60
6016	C3
6017	06
6018	60
6019	7E
601A	EB
601B	4E
601C	77
601D	EB
601E	71
601F	23
6020	C3
6021	06
6022	60
6023	BE
6024	DA

☐ Show entire memory content  
☒ Show only loaded memory location  
☐ Store directly to specified memory location

### b) Descending Order -

The image displays two instances of the 8085 Simulator, each showing a different assembly program and its execution state.

### Left Simulator: C:\Users\Maham Khan\OneDrive\Documents\SELECTION\_SORT\_8085.asm

**Registers:** AX=0000, BX=0000, CX=0000, DX=0000, SP=0000, BP=0000, SI=0000, DI=0000

**Memory Editor:** Memory Range: 0000 - FFFF

Memory Address	Value
0000	00
0001	00
0002	50
0003	46
0004	48
0005	23
0006	7E
0007	E5
0008	23
0009	05
000A	CA
000B	37
000C	60
000D	48
000E	CD
000F	24
0010	60
0011	E1
0012	BE
0013	D2
0014	1A
0015	60
0016	23
0017	C3
0018	06
0019	60
001A	7E
001B	EB
001C	4E
001D	77
001E	EB
001F	71
0020	23
0021	C3
0022	06
0023	60
0024	BE

**Simulation:** Start From → 0000

**Assembly Code:**

```

0000 INITIA LXI H,5000
0001
0002
0003 MOV B,H
0004 MOV C,B
0005 JNZ H
0006 START MOV A,H
0007 PUSH H
0008 JNZ H
0009 DCR B
000A JZ STOP
000B
000C
000D
000E MOV C,B
000F CALL MIN
0010
0011 POP H
0012 CHP H
  
```

### Right Simulator: C:\Users\Maham Khan\OneDrive\Documents\SELECTION\_SORT\_8085.asm

**Registers:** AX=0000, BX=0000, CX=0000, DX=0000, SP=0000, BP=0000, SI=0000, DI=0000

**Memory Editor:** Memory Range: 0000 - FFFF

Memory Address	Value
0000	00
0001	00
0002	50
0003	46
0004	48
0005	23
0006	7E
0007	E5
0008	23
0009	05
000A	CA
000B	37
000C	60
000D	48
000E	CD
000F	24
0010	60
0011	E1
0012	BE
0013	D2
0014	1A
0015	60
0016	23
0017	C3
0018	06
0019	60
001A	7E
001B	EB
001C	4E
001D	77
001E	EB
001F	71
0020	23
0021	C3
0022	06
0023	60
0024	BE

**Simulation:** Start From → 0000

**Assembly Code:**

```

0000 INITIA LXI H,5000
0001
0002
0003 MOV B,H
0004 MOV C,B
0005 JNZ H
0006 START MOV A,H
0007 PUSH H
0008 JNZ H
0009 DCR B
000A JZ STOP
000B
000C
000D
000E MOV C,B
000F CALL MIN
0010
0011 POP H
0012 CHP H
  
```

## 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