OBJECTIVE - 1

Find the Largest/Smallest number (8-bit number) from a given array of size N.

PRE-LAB

Assembly Code

```
•data
Count db 04h; count = array size
Value db 09h, 10h, 05h, 03h; array elements
res db ? ; store the result in res
•Code
Main PROC
        mov ax, data
        mov ds, ax
        mov cx, count
        dec cx
        LEA SI, Value
        mov al, [SI]
up:     inc si
        cmp al, [si]
        jnl nxt ; jump to "nxt" if not less than
        mov al, [si]
nxt:    dec cx
        jnz up
        LEA DI, res
        mov [DI], al
END MAIN
```

Input / Output Analysis

Input:

- An array of 4 elements: 09h, 10h, 05h, 03h (Hexadecimal).
- The Program scans through the array to find the Largest number.

Output:

- The largest number found in the array is 10h (16 in decimal)
- This result is stored in the memory location res.

OBJECTIVE - 2

Arrange the elements (8-bit number) of a given array of size N in ascending / descending order.

PRE-LAB

Assembly Code

```
•Data
Count DB 06
Value DB 09H, 0FH, 14H, 45H, 24H, 3FH
•CODE
MAIN PROC
        MOV AX, DATA
        MOV DS, AX
        LEA DI, count
        MOV CH, [DI]
        DEC CH
UP2:    MOV CL, CH
        LEA SI, Value
UP1:    MOV AL, [SI]
        CMP AL, [SI + 1]
        JC DOWN          ; JNC for Descending
```

```
        MOV DL, [SI+1]
        XCHG [SI], DL        ; swapping of MEMORY location DATA
        MOV [SI+1], DL
DOWN:   INC SI
        DEC CL
        JNZ UP1
        DEC CH
        JNZ UP2
END MAIN
```

Input/Output Analysis

Input:

- The given array (09H, 0FH, 14H, 45H, 24H, 3FH) Consists of six hexadecimal numbers. The task is to arrange these numbers in ascending order using sorting logic.

Output

After sorting, the array is rearranged as (09H, 0FH, 14H, 24H, 3FH, 45H) in ascending order. The sorted values are stored in memory for further use.

OBSERVATION TABLE

OBJ - 1

SL no.	Memory Location	operand
1	0000	04H
2	0001	09H
3	0002	10H
4	0003	05H
5	0004	03H

(Input)

SL no.	Memory Location	Operand
1	0005	10H (Greatest)
2	0005	03H (Smallest)

(output)

OBJ-2

SL no.	Memory Location	operand
1	0000	06H
2	0001	09H
3	0002	0FH
4	0003	14H
5	0004	45H
6	0005	24H
7	0006	3FH

(Input)

SL no.	Memory Location	Operand
1	0000	06H
2	0001	09H
3	0002	0FH
4	0003	14H
5	0004	24H
6	0005	3FH
7	0006	45H

(Output)

Conclusion

The lab exercise on 8086 assembly Language programming focused on array manipulation. Particularly finding the smallest number in an array in ascending order and sorting element of array in descending order. Through practic implementation, we depend our understanding of low level Programming concept and learned essential skills in algorithmic thinking.

POST-LAB

1.) What are the directives available for data declaration in 8086 microprocessors?

Ans:-

(a) Define Byte (DB) -> Define one or more bytes of data.
(b) Define word (DW) -> Define one or more word (16-bit).
(c) Define Double (DD) -> Define one or more double word (32 bit).
(d) Define Quadwords (DQ) -> Define one or more quadwords (64 bits)
(e) Define Ten Bytes (DT) -> Define a variable that is 10 byte.

2.) State the difference b/w END, ENDP, and ENDS directives.

Ans:-

END	ENDP	ENDS
Marks the end of entire program.	Marks the end of a Procedure (Similar to function).	Marks the end of a segment (a logical division of Program).

3.) Find the sum & average of a given array of size N.

Ans:-

• data

```
Array_Length db 04h
Array db 05h, 10h, 05h, 03h
Sum db 00h
Average db 00h
```

• code

```
MAIN PROC:
        mov ax, data
        mov ds, ax
        mov cx, array_Length
        LEA SI, [SI]
```

```
xor bx, bx ; calculate sum
sum_loop : add bl, al
inc si
mov al, [si]
dec cx
jnz sum_loop
mov al, bl ; calculate average
mov bl, array_length
div bl
mov average, al
MAIN ENDP
END MAIN
```

27.03.25