Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: (Fall, Year:2025), B.Sc. in CSE (Day)

Lab Report NO# 04
Course Title: Microprocessors & Microcontrollers Lab
Course Code: CSE 304 Section:232_D2

Lab Experiment Name: Practice of Array in Emu8086

Student Details

	Name	ID
1.	Md. Sohan Millat Sakib	222902036

Lab Date : 10-11-2025
Submission Date : 17-12-2025
Course Teacher's Name : Sagufta Sabah Nakshi

Lab Report Status

Marks: …… **Signature:.....**
Comments:..... **Date:.....**

1st problem

1. TITLE OF THE LAB REPORT EXPERIMENT

Write an Assembly program to take array elements from the user and calculate:

- Average
- Largest number
- Smallest number

2. OBJECTIVES

The objectives of this experiment are:

- To understand how to take multiple inputs using loops in Assembly.
- To calculate sum, largest, and smallest values in an array.
- To practice using procedures (PROC/ENDP) in Assembly.
- To use interrupts for input/output operations.

3. PROCEDURE

i. D Declare an array of 5 elements.
ii. Display message asking user to enter elements.
iii. Take each input using INT 21h (AH = 01) and store inside the array.
iv. Convert ASCII input to actual numeric value by subtracting 48.
v. For each input:
   - Add value to SUM
   - Compare to find LARGEST
   - Compare to find SMALLEST
vi. After inputs are taken:
   - Divide SUM by number of elements to find AVERAGE
   - Print Average, Largest, Smallest
vii. Use separate procedures for summation, max, min, and printing results.

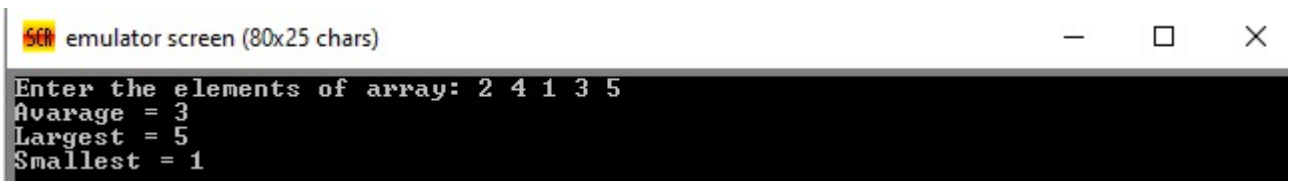
4. IMPLEMENTATION

```
org 100h
.DATA
array db 10 dup(1)
space db " $"
NEWLINE DB 0DH,0AH,"$"
msg1 DB "Enter the elements of array: $"
msg2 DB "Avarage = $"
msg3 DB "Largest = $"
msg4 DB "Smallest = $"
size DW 5
largest DB 0
smallest DB 255
```

```
sum DB 0
.CODE
MAIN PROC
   MOV AX,@DATA
   MOV DS,AX
   MOV AH,09H
   LEA DX,msg1
   INT 21H
   MOV SI,OFFSET array
   MOV CX,size
   read_input:
      MOV AH,01H
      INT 21H
      MOV [SI],AL
      CALL SUMMATION
      CALL LARGESTT
      CALL SMALLESTT
      MOV AH,09H
      LEA DX,space
      INT 21H
      INC SI
      LOOP read_input
      CALL PRINT_ALL
ret
SUMMATION PROC
MOV BL,sum
MOV BH,AL
SUB BH,48
ADD BL,BH
MOV SUM,BL
RET
SUMMATION ENDP
LARGESTT PROC
MOV BL,largest
MOV BH,AL
SUB BH,48
CMP BH,BL
JA update
JMP end_
update:
MOV largest,BH
end_:
RET
LARGESTT ENDP
SMALLESTT PROC
MOV BL,smallest
MOV BH,AL
SUB BH,48
CMP BH,BL
JB update1
JMP end1_
update1:
MOV smallest,BH
end1_:
RET
SMALLESTT ENDP
```

```
PRINT_ALL PROC
;print new line
MOV AH,09H
LEA DX, NEWLINE
INT 21H
MOV AH,09H
LEA DX,msg2
INT 21H
MOV AX,0
MOV AL,sum
MOV BX,size
MOV BH,0
DIV BL
ADD AL,48
MOV AH,02H
MOV DL,AL
INT 21H
MOV AH,09H
LEA DX, NEWLINE
INT 21H
MOV AH,09H
LEA DX,msg3
INT 21H
MOV AL,largest
ADD AL,48
MOV AH,02H
MOV DL,AL
INT 21H
MOV AH,09H
LEA DX, NEWLINE
INT 21H
MOV AH,09H
LEA DX,msg4
INT 21H
MOV AL,smallest
ADD AL,48
MOV AH,02H
MOV DL,AL
INT 21H
RET
PRINT_ALL ENDP
```

5. TEST RESULT



6. ANALYSIS AND DISCUSSION

During write this program I don’t face any kind of issue. No problem at all

2nd Problem:

1. TITLE OF THE LAB REPORT EXPERIMENT

Write an Assembly program to sort an array using Bubble Sort in:

- Ascending order
- Descending order

2. OBJECTIVES

The main objectives of this experiment are:

- To understand how to compare and swap elements in Assembly.
- To implement Bubble Sort algorithm using loops.
- To print sorted values using INT 21h.
- To practice working with arrays and index registers (SI).

3. PROCEDURE

1. Declare variable A to store the number.
2. Take user input and store it in CX (loop counter).
3. Print a new line before showing output.
4. Set **DX = 0**, **BX = 0**, **AX = 1** to start factorial calculation.
5. Each loop:
6. Increase A
7. Increase BX
8. Multiply AX by A and store result
9. After loop ends, divide and print digits one by one.

4. IMPLEMENTATION

```
.model small
.stack 100h
.data
    space db " $"
    msg1 db 10,13,"Enter the elements of array: $"
    msg2 db 10,13,"Ascending: $"
    msg3 db 10,13,"Descending: $"
    arr  db 7 dup(?)
    n    db 7
.code
main proc
    mov ax, @data
    mov ds, ax
    call INPUT_ARRAY
    call BUBBLE_ASC
    lea dx, msg2
    mov ah, 09h
    int 21h
```

```
    call PRINT_ARRAY
    call BUBBLE_DESC
    lea dx, msg3
    mov ah, 09h
    int 21h
    call PRINT_ARRAY
    mov ah, 4Ch
    int 21h
main endp
INPUT_ARRAY proc
    lea dx, msg1
    mov ah, 09h
    int 21h
    mov cx, 7
    mov si, 0

IN_LOOP:
    mov ah, 01h
    int 21h
    sub al, '0'
    mov arr[si], al
    inc si
    mov ah,09h
    lea dx,space
    int 21h
    loop IN_LOOP
    ret
INPUT_ARRAY endp
BUBBLE_ASC proc
    mov cl, n
    dec cl
OUTER1:
    mov si, 0
    mov ch, cl
INNER1:
    mov al, arr[si]
    mov bl, arr[si+1]
    cmp al, bl
    jbe NO_SWAP1
    mov arr[si], bl
    mov arr[si+1], al
NO_SWAP1:
    inc si
    dec ch
    jnz INNER1
    dec cl
    jnz OUTER1
    ret
BUBBLE_ASC endp
BUBBLE_DESC proc
    mov cl, n
    dec cl
OUTER2:
    mov si, 0
    mov ch, cl
INNER2:
    mov al, arr[si]
    mov bl, arr[si+1]
    cmp al, bl
    jae NO_SWAP2
    mov arr[si], bl
    mov arr[si+1], al
```

```
NO_SWAP2:
    inc si
    dec ch
    jnz INNER2
    dec cl
    jnz OUTER2
    ret
BUBBLE_DESC endp
PRINT_ARRAY proc
    mov cx, 7
    mov si, 0
PRINT_LOOP:
    mov dl, arr[si]
    add dl, '0'
    mov ah, 02h
    int 21h
    mov dl, ' '
    int 21h
    inc si
    loop PRINT_LOOP
    ret
PRINT_ARRAY endp
end main
```

5. TEST RESULT

6. ANALYSIS AND DISCUSSION

- Bubble sort works correctly in both directions.
- Input and output formatting are clear.
- Efficiency of Bubble Sort is low, but implementation is simple for 8086.
- The program demonstrates understanding of loops, comparisons, and swapping.

SUMMARY:

In this lab, I practiced handling arrays in 8086 Assembly by taking user inputs, finding the average, largest, and smallest values, and sorting the array in ascending and descending order using bubble sort. Through these tasks, I learned how loops, comparisons, and procedures work in Assembly and improved my understanding of basic microprocessor operations.