Computer Organization and Architecture (EET2211)

LAB III: Analyze and Evaluate the Array Operations using 8086 microprocessors.

Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar

Branch: Computer Science & Engineering		**Section:** 2241044	
S. No.	**Name**	**Registration No.**	**Signature**
27	E. Jagadeeswar Patro	2241016309	E. Jagadeeswar Patro

Marks:_____/10

Remarks:

Teacher's Signature

I. OBJECTIVE:

1. Find the largest/smallest number (8-bit number) from a given array of size N.
2. Arrange the elements (8-bit number) of a given array of size N in ascending/descending order.

II. PRE-LAB

Objective 1:

Finding largest number(8-bit):

```
.data                    ; data segment
   count db 04h          ; count = array size, offset value = [0000 h]
   value db 09h, 10h, 05h, 03h ; array elements [0001h] to [0004h]
   res db 0              ; store result in res
.code                    ; code segment
MAIN PROC
   mov ax, data          ; store data to ax
   mov ds, ax            ; store value of ax in ds
   mov cl, count         ; cl = 04h
   dec cl                ; decrement once before loop as N-1 iteration required
   LEA SI, value         ; address of 'value' set into 'SI'
   mov al, [SI]          ; move 1st value of array int al
up: inc si               ; Increment SI, Array index = 2nd
   cmp al, [si]          ; compare 1st and 2nd element of array and update al
   jnc nxt               ; jump if not carry
   mov al, [si]          ; if less, move value of si into al
nxt: dec cl              ; decrement count
   jnz up                ; jump if cl not equal to 0
   LEA DI, res           ; effective address of res
   mov [DI], al          ; value of al is stored at res
END MAIN
```

- **Finding smallest number(8-bit) from array of size N:**

- Data

```
count db 04h ; array size
value db 09h, 10h, 05h, 03h ; array elements
res db 0 ; result address
```

- code

```
MAIN PROC
    mov ax, data    ; store data's offset address to ax
    mov ds, ax      ; store value of ax in ds
    mov cl, count   ; cl = 04h
    dec cl          ; dec once for N-1 iterations
    LEA SI, value   ; address of value array set into SI
    mov al, [SI]    ; move 1st element into al

up: inc si          ; Array Index = 2nd
    cmp al, [si]    ; Compare 1st & 2nd element
    jc nxt          ; if carry, jump
    mov al, [si]    ; else move value of si into al

nxt: dec cl         ; decrement count
    jnz up          ; jump if cl not equal to 0
    LEA DI, res     ; address of res loaded
    mov [DI], al ; result stored.

END MAIN
```

Objective 2:

- Arranging in Ascending Order:

```
.data
    count db 06 ; array size
    value db 09h, 0Fh, 14h, 24h, 3Fh ; array

.code
MAIN PROC
    mov ax, data ; store data into ax register
    mov ds, ax ; store also into ds
    mov ch, count ; set ch = count
    dec ch ; decrementing once for N-1 operation.
UP1: mov cl, ch ; store the current index
    lea si, value ; effective address of 'value'
UP2: mov al, [SI] ; 1st element into al
    cmp al, [SI+1] ; compare with 2nd element
    jc DOWN ; jump if carry
    mov dl, [SI+1] ; 2nd value into dl
    xchg [SI], dl ; exchange 1st and 2nd
    mov [SI+1], dl ; store dl into 2nd position
DOWN: inc si ; increment index for next iteration
    dec cl ; decrease counter by 1
    jnz UP2 ; loop until counter becomes zero
    dec ch ; decrement count
    jnz UP1 ; jump if cl ≠ 0

END MAIN
```

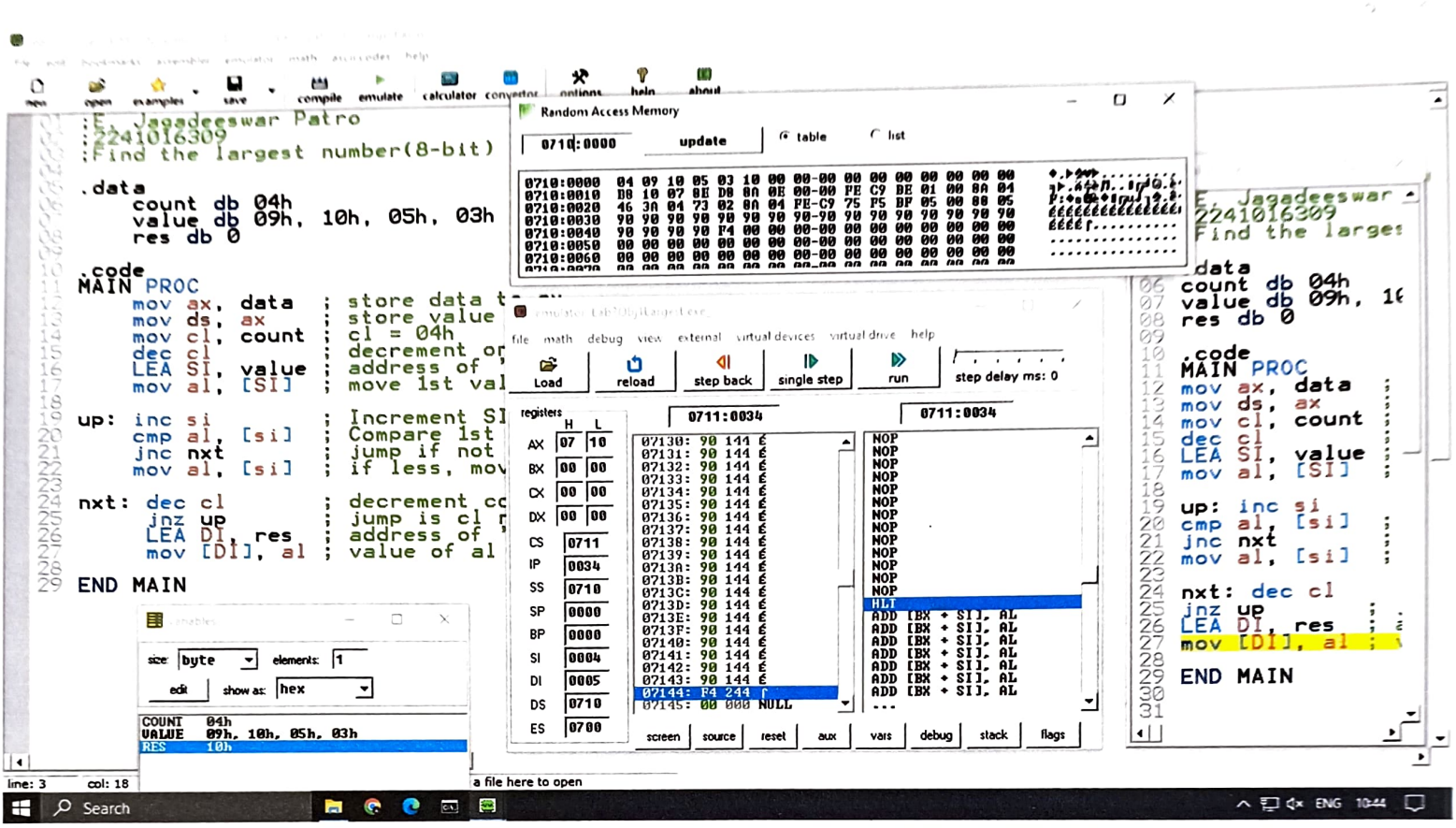
- **Arranging in Descending Order:**

```
.data
    count db 06 ; array size
    value db 09h, 0Fh, 14h, 45h, 24h, 3Fh ; array

.code
MAIN PROC
    mov ax, data ; offset address of data segment
    mov ds, ax ; copied to ds
    mov ch, count ; ch = count = 06h
    dec ch ; decrement one before loop (N-1 iterations required)

UP2: mov cl, ch ; starting index value
     lea si, value ; its effective address

UP1: mov al, [si] ; 1st value of array in al
     cmp al, [si+1] ; compare with second value
     jnc DOWN ; jump if not carry
     mov dl, [si+1] ; exchange
     xchg [si], dl ; procedure
     mov [si+1], dl ; if 1st < 2nd

DOWN: inc si ; increment index for next
      dec cl ; count--
      jnz UP1 ; loop until counter = 0
      dec ch ; count--
      jnz UP2 ; jump if cl ≠ 0

END MAIN
```

III. LAB : -

Objective 1:

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



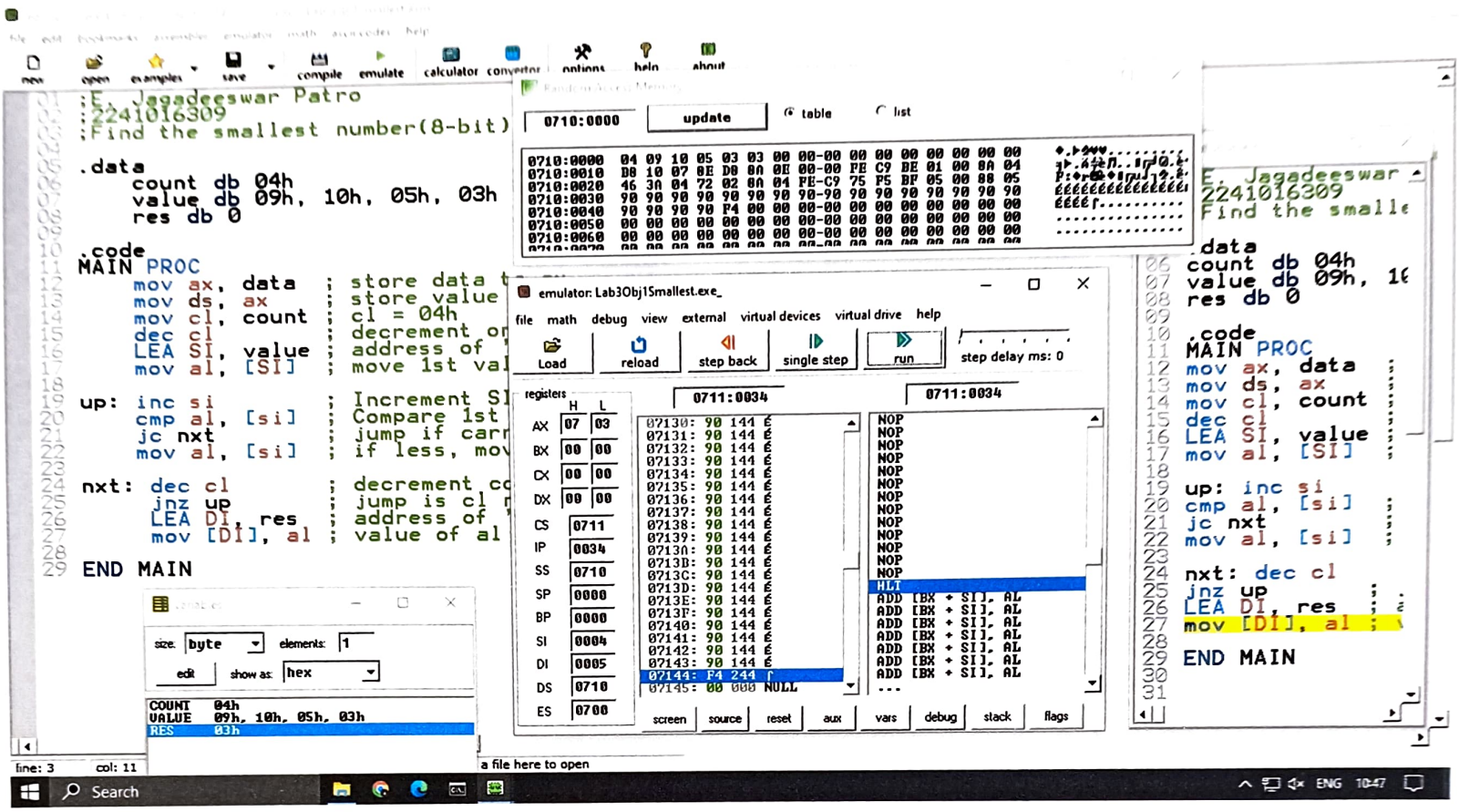
Input:

Sl. No.	Memory Location	Operand (Data)
1	0710:0000	04h
2	0710:0001	09h
3	0710:0002	18h
4	0710:0003	05h
5	0710:0004	03h

Output:

Sl. No.	Memory Location	Operand (Data)
1	0210:0005	18h

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



Input:

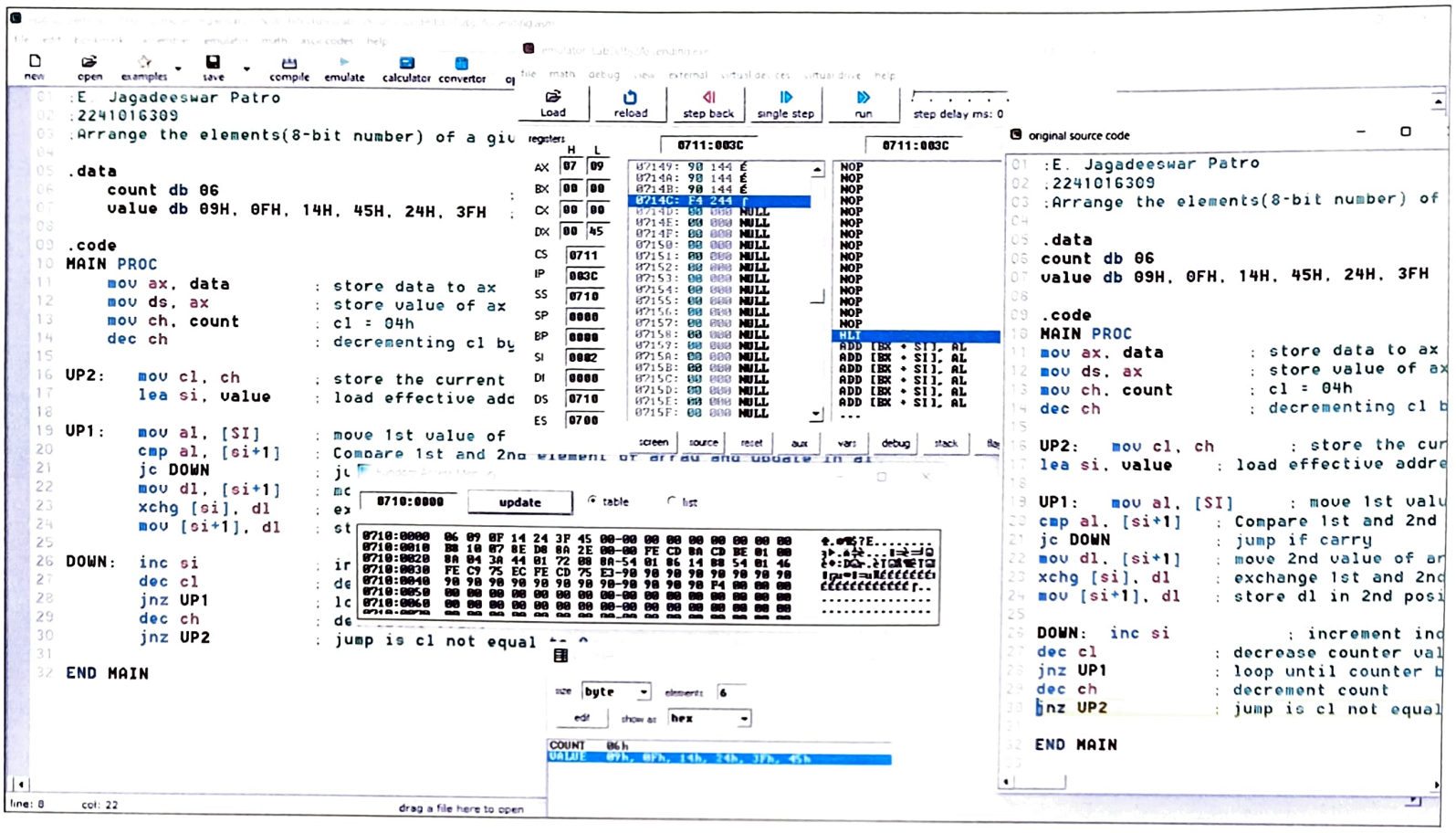
Sl. No.	Memory Location	Operand (Data)
1	0710:0000	04h
2	0710:0001	09h
3	0710:0002	10h
4	0710:0003	05h
5	0710:0004	03h

Output:

Sl. No.	Memory Location	Operand (Data)
1	0710:0005	03h

Objective 2:

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



From this result, I have observed.....

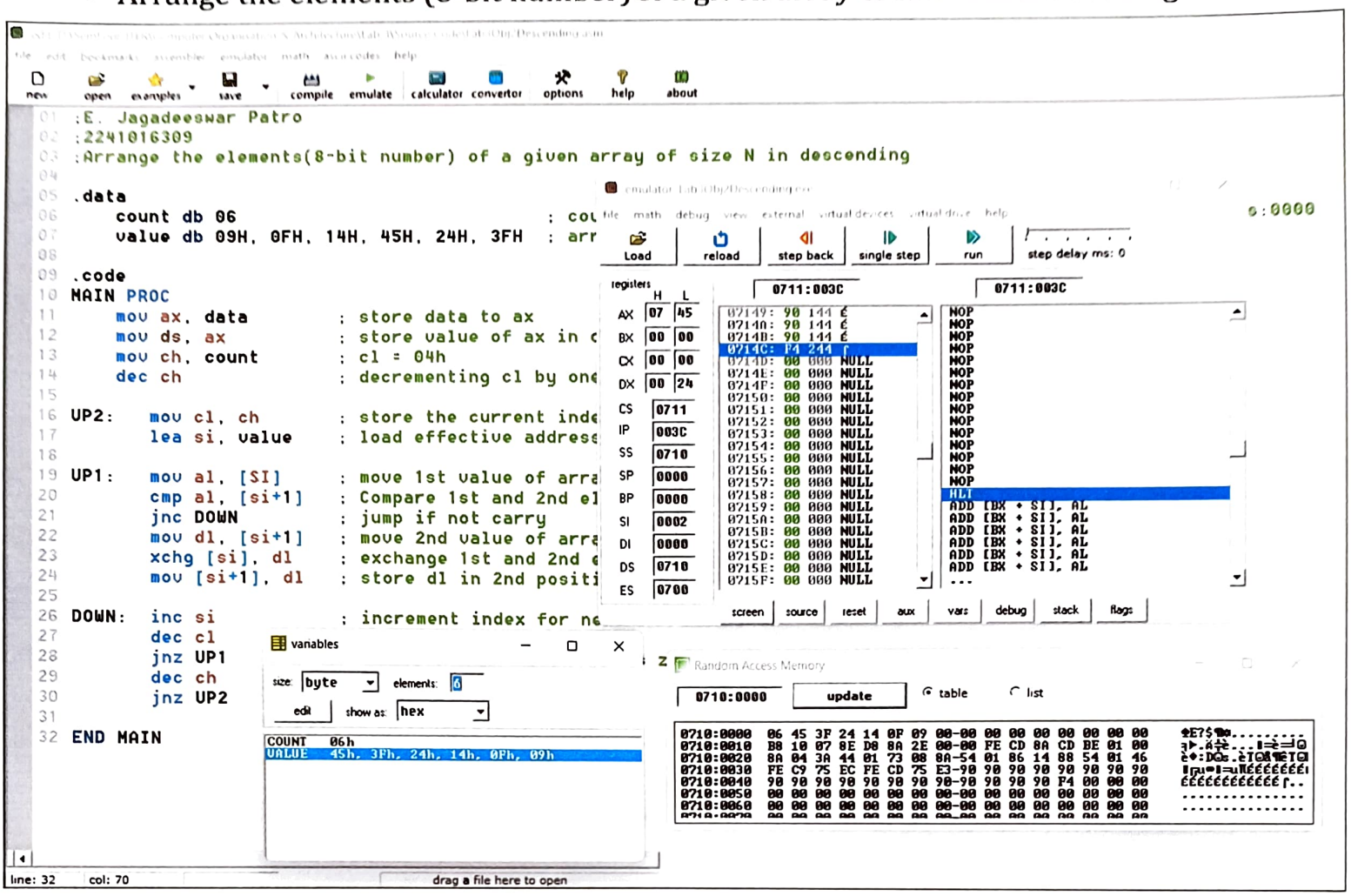
Input:

Sl. No.	Memory Location	Operand (Data)
1	0710:0000	06h
2	0710:0001	09h
3	0710:0002	0Fh
4	0710:0003	14h
5	0710:0004	45h
6	0710:0005	24h
7	0710:0006	3Fh

Output:

Sl. No.	Memory Location	Operand (Data)
1	0710:0001	09h
2	0710:0002	0Fh
3	0710:0003	14h
4	0710:0004	24h
5	0710:0005	3Fh
6	0710:0006	45h

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



From this result, I have observed.....

Input:

Sl. No.	Memory Location	Operand (Data)
1	0710:0000	06h
2	0710:0001	09h
3	0710:0002	0Fh
4	0710:0003	14h
5	0710:0004	45h
6	0710:0005	24h
7	0710:0006	3Fh

Output:

Sl. No.	Memory Location	Operand (Data)
1	0710:0000	45h
2	0710:0002	3Fh
3	0710:0003	24h
4	0710:0004	14h
5	0710:0005	0Fh
6	0710:0006	09h

IV. CONCLUSION

The lab exercise on 8086 assembly language programming focused on array manipulation, particularly finding smallest / largest element of an array and arranging an array in ascending / descending order. Through practical implementation, we deepend our understanding of low-level programming concepts and learned essential skills in algorithmic thinking & precise coding practices for 8086 microprocessor.

V. POST LAB

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

A) Directives available are:

i) DB (Define Byte) - Defines one are more bytes of data.

ii) DW (Define Word) - Defines one or more words (16-bit data)

iii) DD (Define Doubleword) - Defines 1 or more double words (32-bit data)

iv) DQ (Define Quadword) - Defines one or more quadwords (64-bit data)

v) DT - Defines a variable that is 10 bytes.

vi) CRLF = Defines a carriage return, or terminator byte.

2. State the difference between END, ENDP, and ENDS directives.

A) END: Marks the end of the entire program

ENDP: Marks the end of a procedure (similar to a function)

ENDS: Marks the end of a segment (a logical division of the program)

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

A) .data

```
count db 04h    ; array length
arr1 db 09h, 10h, 05h, 03h ; array elements
sum db 00h      ; store sum
avg db 00h      ; store average
```

.code

```
MAIN PROC
    mov ax, data    ; initialize data segment
    mov ds, ax
    mov ch, count   ; CH = 04h
    xor bx, bx      ; clear bx register
    mov cl, ch      ; copy count into cl
    lea si, arr1    ; point to 1st element of arr1
SUMM: mov al, [SI]  ; current element stored in al
    add bl, al      ; al + bl in [bl]
    inc si          ; increment pointer
    dec cl          ; decrement counter
    jnz SUMM        ; if CX ≠ 0 jump back to 'SUMM' loop
    lea di, sum     ; effective address of 'sum'
    mov [di], bl    ; store sum
    mov ax, 0000h   ; reset ax register
    mov al, bl      ; store sum into al
    mov bl, count   ; store count into bl
    div bx          ; ax / bx
    lea di, avg     ; effective address of 'avg'
    mov [di], al    ; store average into 'avg'
END MAIN
```