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: CSE		Section: 2241026	
S. No.	**Name**	**Registration No.**	**Signature**
6	Dinanath Dash	2241004161	Dinanath Dash

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

For each objective in prelab describe the following points:

- Write the assembly code with a description (ex. Mov ax,3000h – ax<-3000h)

For Obj:1a →

```
.data
count db, 04h; define variable count to store array size
value db, 09h, 10h, 05h, 03h; define array value with elements
res db 0; define variable res to store the result

.code
Main Proc; start of the main procedure
mov ax, data; Move the address of the data segment to ax
    mov ds, ax; Set the data segment to ds register
    mov cl, count; Move the value of count to cl
    dec cl; Decrement cl by 1
    LEA SI ;value; Load effective address of value into SI
    mov al , [SI] ; move the first element of value to al

up:
    inc si; increment si
    cmp al; compare the current element in AL with the next element in the array
    jnl nxt; jump to nxt if no less than or equal
    mov al;[SI]; move to the next element to al if it's greater
nxt:
    dec cl ; decrement cl
    jnz up ; jump to up if counter is not zero
    lea di; load effective address of res into di
    mov [di], al; move the result (stored in AL) to the 'res' variable
end main : end of main procedure
```

Teacher's Signature.....

For Obj. 1b:

```
.data
count db 04h;
Value db 09h, 10h, 05h,03h;
res db 0;
.code
Main Proc;
  mov ax, data;
  mov ds, ax;
  mov cl, count;
  dec cl;
  lea si;
  mov al, [si];
up:
   inc si;
   Cmp al
   jl nxt ; Jump to nxt if less than or equal
   mov al
nxt:
   dec cl
   jnz up
   lea di, res
  mov [di], al
end main
```

For Obj. 2a:

```
.data
count db 06; define variable count to store array size
value db 09h, 0Fh, 14h, 45h, 24h, 3Fh; define array 'value' with elements
.code
main proc; Start of the main procedure
    move ax, data; move the ~~data~~ address of the data segment to ax
    mov ds, ax; Set the data segment to ds register
    mov ch,count; Move the value of count to ch
    dec ch; decrement ch
up2: mov cl,ch; move the value of ch to cl
     lea si, value; Load effective address of 'value' array into SI
up1: mov al,[si]; move the value at address pointed by si to al
     cmp al, [si+1]; compare al with the next element in the array
```

```
    jc down; jump to down if al with the next element in the array
    mov dl, [si+1] ; move the next element to dl
    xchg [si], dl; exchange the values of al and dl
    mov [si+1], dl ; mov the value of dl back to the array
down:
    inc si; increment si
    dec cl; decrement cl
    jnz up1; jump to up1 if cl is not zero
    dec ch; decrement ch
    jnz up2 ; jump to up2 if ch is not zero
end main;
```

For Obj. 2b):

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

.code
main proc
        mov ax, data
        mov de, [illegible]ax;
        mov [illegible] ch;
        dec, ch;
up2:    mov cl, ch
        lea si
up1:    mov al, [si];
        cmp al, [si+1]
        jnc down; jump to down if al [illegible] is not less than the next element
        mov dl, [si+1];
        xchg [si], dl;
        mov [si+1], dl
down:   inc si;
        dec cl;
        jnz up1
        dec ch
        jnz up2
end main;
```

From the results, I have observed....

Input:

Output:

Sl. No.	Memory Location	Operand (Data)	Sl. No.	Memory Location	Operand (Data)
1a	0000h	04	1a	0005h	10
	0001h, 0002h, 0003h, 0004h	09, 10, 05, 03			
1b	0000h	04	1b	0005h	03
	0001h, 0002h, 0003h, 0004h	09, 10, 05, 03			
2a	0000h	06	2a	0001h, 0002h, 0003h, 0004h, 0005h, 0006h	09, 0F, 14, 24, 3F, 45
	0001h, 0002h, 0003h, 0004h, 0005h, 0006h	09, 0F, 14, 45, 24, 3F			
2b	0000h	06	2b	0001h, 0002h, 0003h, 0004h, 0005h, 0006h	45, 3F, 24, 14, 0F, 09
	0001h, 0002h, 0003h, 0004h, 0005h, 0006h	09, 0F, 14, 45, 24, 3F			

Conclusion :- In conclusion, the experiment conducted in Emu8086 successfully demonstrated the implementation of basic array ~~obo~~ operations: finding the largest / smallest number and arranging elements in ascending / descending order. Through practical implementation and analysis, we gained insights into fundamental concepts such as iteration, comparison and sorting algorithms.

V) Post Lab →

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

Ans → In 8086 assembly language programming, directives are used to declare and define various types of data.

i) DB (Define Byte) : This directive is used to define one or more bytes of data.

ii) DW (Define Word): This directive is used to define one or more (16-bit data).

iii) DD (Define Doubleword): This directive is used to define one or more ~~quad~~ double words (32-bit data).

iv) DQ (Define Quadword) : This directive is used to define one or more quad words (64-bit data).

v) DT (Define Ten bytes): This directive is used to define ten bytes of data, typically used for defining packed decimal values.

vi) DS (Define String) : This directive is used to define a string of characters.

vii) RESB, RESW, RESD, RESQ, REST : These derivates are used to reserve space for a specified number of bytes, words, doublewords, quadwords or ten-bytes data.

Q2) State the difference between END, ENDP, and ENDS directives.

Ans → i) End directives →

a) This marks the end of the source file or program. It typically appears at the end of the main source file or program file.

b) When the assembler encounters the 'END' directive, it signifies the end of the assembly process for that file.

ii) ENDP directives →

a) This marks the end of a procedure definition in assembly language.

b) It is used in conjuction with the 'PROC' directive, which starts the definition of a procedure.

c) When the assembler encounters the 'ENDP' directive, it signifies the end of the procedure definition.

iii) ENDS directives →

a) This marks the end of a segment definition in assembly language.

b) It is used to terminate a segment block that was started with the 'segment' directive.

c) When the assembler encounters the 'ENDS' directive, it signifies the end of the ~~procedure~~ segment definition.