

Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year: 2021), B.Sc. in CSE (Day)

Lab Report No: 05

Course Title: Microprocessors and Microcontrollers Lab

Course Code: CSE 304 Section: PC-DD

Lab Experiment Name: Introduction of understanding the use of Procedure.

Student Details

Name	Id
Md. Romzan Alom	201902144

Lab Date: 29.11. 2021

Submission Date: 06.12.2021

Course Teacher's Name: Rusmita Halim Chaity

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status	
Marks:	Signature:.....
Comments:.....	Date:.....

Title of the Lab Experiment: Introduction of understanding the use of Procedure.

Objectives / Aim:

We learn about array from this experiment. We can take user input as elements of array index and we can perform various operations on this input and show the result as output. Look like sorting an array.

Procedure / Analysis / Design:

Instruction MOV AX, BX is used moves data BX to AX and data is stored in AX.

Instruction DIV BX is used division. Here, AX is divided by BX and Quotient is stored in AL and Remainder is stored in AH.

Instruction ADD AX, BX is used addition. Here, add to BX and AX and this data is stored in AX.

Instruction SUB AX, BX is used subtraction. Here, BX and AX are subtracted and this data is stored in AX.

Instruction MOV AH, 1 and INT 21H are used for user input and it stored AX.

Instruction MOV DX, AX and MOV AH, 2 and INT 21H are used for show output. Here, DX store AX data and compiler understand DX data then it show DX data as output.

Instruction LEA DX, R and MOV AH, 9 and INT 21H are used for call variable and show output the variable data.

Instruction MOV AH,2 and MOV DL,0DH and INT 21H and MOV DL,0AH and INT 21H are used for newline.

Instruction MOV AH,2 and MOV DL,0DH and INT 21H and MOV DL,0AH and INT 21H are used for newline.

Instruction Array DB N (?) are used for initialize an array.

Instruction CMP AX, BX are used for compare AX and BX.

Instruction JL and JGE are used for CMP condition.

Instruction RESULT_1: are created a level.

Instruction LOOP START is used for looping. It goes to START level and this level decrease with CX.

Instruction R PROC to R ENDP is used for procedure.

Instruction call is used for call procedure.

Instruction V DB 0AH, 0DH, "The result is: \$" are used for newline and take a message. Here, 0AH, 0DH, is used newline.

Problem-01

1. Write an Assembly Language code that takes any 5 of decimal digits (0 9) as input and calculates the average, largest and smallest of them in three different procedures and show the output like the following:

Input:

Enter he elements of array: 2 4 1 3 5

Output:

AVERAGE=3

LARGEST=5

SMALLEST = 1

Pseudo-code:

.MODEL SMALL

.STACK 100H

.DATA

n db ?

Av db ?

Lz db ?

Sm db ?

R db ?

A db n dup (?)

R1 DB "Enter the number of input: \$"

R2 DB 0AH,0DH,"Enter the all element of index: \$"

R3 DB 0AH,0DH,"The result of our program: \$"

R4 DB 0AH,0DH,"\$"

R5 DB 0AH,0DH,"The average number of array: \$"

R6 DB 0AH,0DH,"The largest of number of array: \$"

R7 DB 0AH,0DH,"The smallest of number of array: \$"

.CODE

MAIN PROC

mov ax, @DATA

mov ds, ax

mov Av,0

mov Lz,0

mov Sm,0

mov R,0

lea dx,R1

mov ah,9

int 21H

mov ah,1

int 21H

sub al,30H

mov n,al

lea dx,R4

mov ah,9

int 21H

lea dx,R2

mov ah,9

int 21H

xor cx,cx

mov cl,n

mov si,0

Loop_1:

mov ah,1

int 21H

sub al,30H

add Av,al

mov A[si],al

inc si

loop Loop_1

lea dx,R4

mov ah,9

int 21H

lea dx,R3

mov ah,9

int 21H

lea dx,R4

mov ah,9

int 21H

lea dx,R5

mov ah,9

int 21H

call Average

mov ah,2

mov dl,Av

add dl,30H

int 21H

```
xor cx,cx
```

```
mov cl,n
```

```
mov si,0
```

```
Loop_2:
```

```
mov ax,00
```

```
mov al,Lz
```

```
cmp A[si],al
```

```
jl RESULT_1
```

```
jge RESULT_2
```

```
RESULT_1:
```

```
mov Lz,al
```


inc si

jmp end1

RESULT_2:

call Larger

jmp end1

end1:

loop Loop_2

xor cx,cx

mov cl,n

mov si,0

mov al,A[si]

mov Sm,al

Loop_3:

mov al,Sm

cmp A[si],al

j1 RESULT_3

jge RESULT_4

RESULT_3:

call Smaller

jmp end2

RESULT_4:

mov Sm,al

inc si

jmp end2

end2:

loop Loop_3

lea dx,R4

mov ah,9

int 21H

lea dx,R6

mov ah,9

int 21H

mov ah,2

mov dl,Lz

add dl,30H

int 21H

lea dx,R4

mov ah,9

int 21H

lea dx,R7

mov ah,9

int 21H

mov ah,2

mov dl,Sm

add dl,30H

int 21H

mov ah, 4ch

int 21h

MAIN ENDP

Average PROC

xor ax,ax

mov al,Av

div n

```
mov Av,al
```

```
ret
```

Average ENDP

Larger PROC

```
mov al,A[si]
```

```
mov Lz,al
```

```
inc si
```

```
ret
```

Larger ENDP

Smaller PROC

```
mov al,A[si]
```

```
mov Sm,al
```

```
inc si
```

ret

Smaller ENDP

END MAIN

Problem-02

Write an Assembly Language code that takes any 7 of decimal digits (0 9) in any order as input and rearrange them in ascending and descending order. Use two different procedures for arranging the digits in ascending and descending order.

Input:

Enter the elements of array: 2 4 1 3 5 9 8

Output:

Ascending: 1 2 3 4 5 8 9

Descending: 9 8 5 4 3 2 1

Pseudo-code:

.MODEL SMALL

.STACK 100H

.DATA

n db ?

R db ?

A db n dup (?)

R1 DB "Enter the number of input: \$"

R2 DB 0AH,0DH,"Enter the all element of index: \$"

R3 DB 0AH,0DH,"The result of our program: \$"

R4 DB 0AH,0DH,"\$"

R5 DB 0AH,0DH,"The ascending order of array: \$"

R6 DB 0AH,0DH,"The descending order of array: \$"

.CODE

MAIN PROC

mov ax, @DATA

mov ds, ax

mov R,0

lea dx,R1

mov ah,9

int 21H

mov ah,1

int 21H

sub al,30H

mov n,al

lea dx,R4

mov ah,9

int 21H

lea dx,R2

mov ah,9

int 21H

xor cx,cx

mov cl,n

mov si,0

Loop_1:

mov ah,1

int 21H

sub al,30H

mov A[si],al

inc si

loop Loop_1

lea dx,R4

mov ah,9

int 21H

lea dx,R3

mov ah,9

int 21H

lea dx,R4

mov ah,9

int 21H

lea dx,R5

mov ah,9

int 21H

xor cx,cx

mov cl,n

mov R,cl

Sort:

call Ascending

mov cl,R

loop Sort

xor cx,cx

mov cl,n

mov si,0

mov di,0

Loop_P:

xor dx,dx

mov ah,02h

mov dl,A[si]

add dl,30H

int 21h

inc si

inc di

loop Loop_P

lea dx,R4

mov ah,9

int 21H

lea dx,R6

mov ah,9

int 21H

call Decending

mov ah, 4ch

int 21h

MAIN ENDP

Ascending PROC

xor cx,cx

mov cl,n

dec cl

mov si,0

Loop_2:

mov al,A[si]

mov bl,A[si+1]

cmp bl,al

jl X1

jge X2

X1:

mov A[si],bl

mov A[si+1],al

jmp end1

X2:

jmp end1

end1:

inc si

loop Loop_2

dec R

mov cl,R

ret

Ascending ENDP

Decending PROC

```
xor cx,cx
```

```
mov cl,n
```

```
dec di
```

```
Loop_P1:
```

```
xor dx,dx
```

```
mov ah,02h
```

```
mov dl,A[di]
```

```
add dl,30H
```

```
int 21h
```

```
dec di
```

```
loop Loop_P1
```

```
ret
```

```
Decending ENDP
```

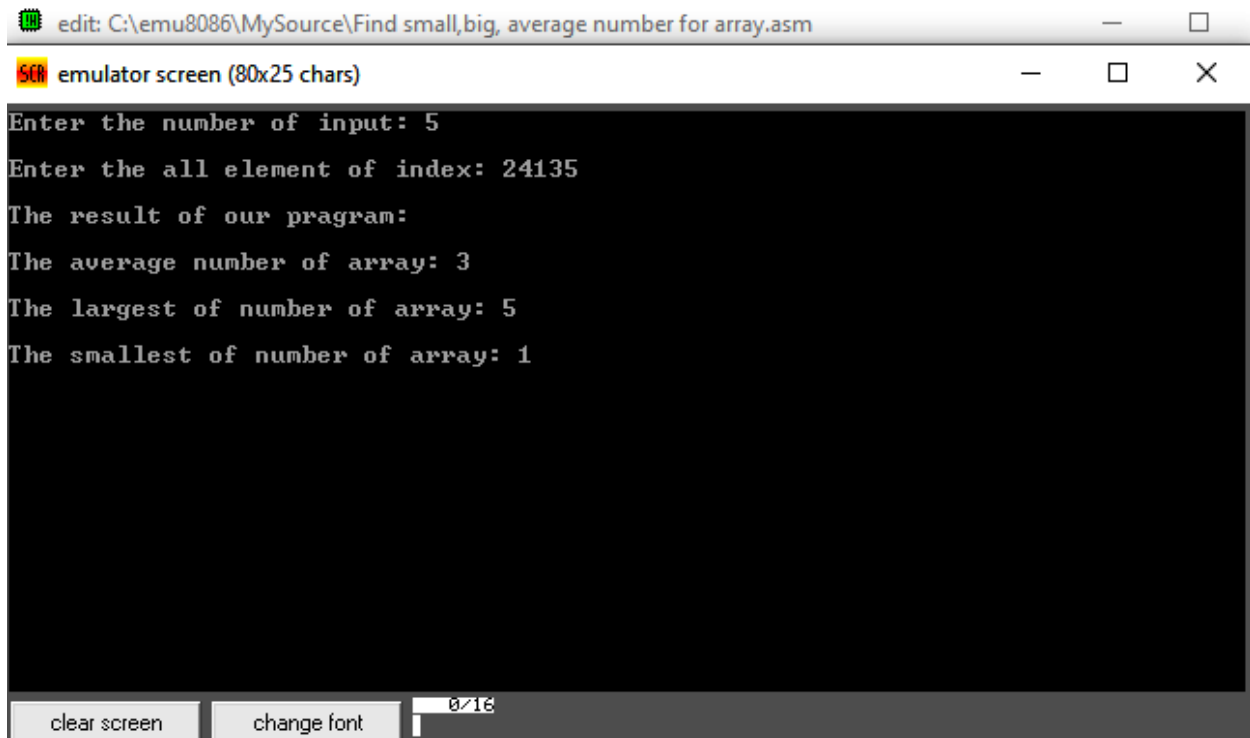
```
END MAIN
```

Calculation:

We will try to find the average, larger, smaller and ascending or descending order of an array. For find average we use a law. That is, average = summation of all elements value / number of elements. On the other hand we use loop to find others.

Test Result / Output:

Problem-1 output:



```
edit: C:\emu8086\MySource\Find small,big, average number for array.asm
emulator screen (80x25 chars)
Enter the number of input: 5
Enter the all element of index: 24135
The result of our program:
The average number of array: 3
The largest of number of array: 5
The smallest of number of array: 1
clear screen change font 0/18
```

The serial numbers are 2, 4, 1, 3 and 5.

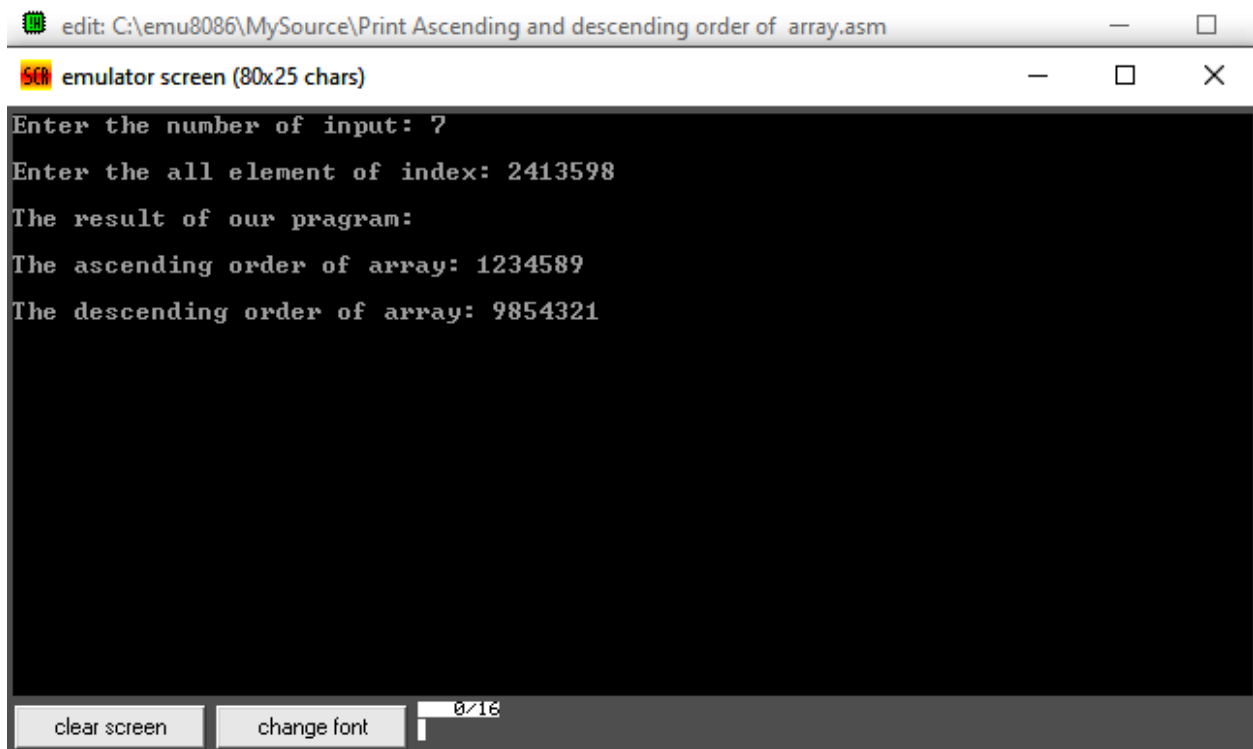
The average value is 3 and it should have been = $(2+4+1+3+5)/5 = 3$.

The larger value is 5 and it should have been = 5.

The smaller value is 1 and it should have been = 1.

We see our output and calculation output are same.

Problem-2 output:



The screenshot shows a window titled "edit: C:\emu8086\MySource\Print Ascending and descending order of array.asm". Below it is an "emulator screen (80x25 chars)" window. The screen displays the following text:

```
Enter the number of input: 7
Enter the all element of index: 2413598
The result of our program:
The ascending order of array: 1234589
The descending order of array: 9854321
```

At the bottom of the emulator screen, there are two buttons: "clear screen" and "change font", and a small status bar showing "0/16".

The serial numbers are 2, 4, 1, 3, 5, 9 and 8.

The ascending order of array is 1, 2, 3, 4, 5, 8, 9 and it should have been 1, 2, 3, 4, 5, 8, 9.

The descending order of array is 9, 8, 5, 4, 3, 2, 1 and it should have been 9, 8, 5, 4, 3, 2, 1.

We see our output and calculation output are same. So, it is 100% right outputs.

Analysis and Discussion:

1. Due to Covid-19 situation, we can't do this experiment directly. So, it is completely based on software.
2. Since, it is done with Software. So it may have some Software and Mechanical errors.

3. To emulate those codes I am facing so many problems for hexadecimal number.
4. From those problems, we use an array. When we operate that array index, we facing some many problem.
5. We can use loop but loop decreases with CX register. That is really confusing to think when loop will stop.
6. To sorting an array I am facing too many problems like swap value in array index.
7. To find average we can find 100% right value because our compiler can't allow any fractional number.
8. From our compile show only char value of hexadecimal number. That why we face some problem to sort array index.
9. Network problem. Cause of the ups and downs of the internet we could not attend the class properly. That why I saw the class recording but this video was very bad quality. This reason I have so many confusion.

SUMMARY:

From this problem, we can take so many decimal numbers from user as input and also take an array from user and print the average, larger, smaller value of array or perform various operations on this input and show the result as output very easily which will help the user to change the program's output. We can also sort some data like array elements and sort them as ascending or descending order. Those are very important to complete this course.