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

Faculty of Sciences and Engineering

Semester: (Fall, Year:2023), B.Sc. in CSE (Day)

LAB REPORT NO: 06

Course Title: Compiler Lab

Course Code: CSE-304 Section:213-D1

Lab Experiment Name: Implement Procedure in Assembly Language Programming.

Student Details

	Name	ID
1.	Irteja Mahmud	213902016

Lab Date : 23-11-2023

Submission Date : 03-12-2023

Course Teacher's Name : Sudip Ghoshal

Lab Report Status

Marks: …… **Signature:.....**

Comments:..... **Date:.....**

1. TITLE OF THE LAB EXPERIMENT

Implement Procedure in Assembly Language Programming.

2. OBJECTIVES/AIM

To reinforce fundamental concepts in assembly language programming, such as loops, input/output operations, branching, conditional checks and procedure.

3.PROCEDURE / ANALYSIS / DESIGN

Problem 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.

Step 1: Initialize the program.

- Set the program's origin to 100h.
- Allocate stack space of 100h.
- Define the model as small.

Step 2: Define data section.

- Declare an array 'digits' to store 5 digits.
- Define messages for input, average, largest, and smallest.
- Declare variables 'len' for array length, 'average', 'largest', and 'smallest'.

Step 3: Start of the code section.

- Begin the 'code' section.

Step 4: Define the 'main' procedure.

- Move data segment address to AX and DS.
- Call the 'initialize' procedure.
- Call the 'average_cal' procedure.
- Call the 'calculate' procedure.
- Call the 'display_output' procedure.
- Set AH to 4Ch and trigger interrupt 21h to terminate the program.

Step 5: Define the 'initialize' procedure.

- Load the address of the input message to DX and display it.
- Initialize a loop to read 5 digits from the user.
- Convert each character to a numerical value and store it in the 'digits' array.

- Increment the array index.
- Display a space character.
- Repeat the loop until 5 digits are read.

Step 6: Define the 'average_cal' procedure.
- Initialize variables for the loop and the sum of digits.
- Loop through the 'digits' array, adding up the values.
- Calculate the average by dividing the sum by the array length.
- Store the result in the 'average' variable.

Step 7: Define the 'calculate' procedure.
- Initialize variables for the loop, largest, and smallest.
- Loop through the 'digits' array.
- Compare each digit with the current largest and smallest.
- Update the largest and smallest accordingly.
- Move the final values to 'largest' and 'smallest'.

Step 8: Define the 'display_output' procedure.
- Display the average, largest, and smallest values along with appropriate messages.

Step 9: End the 'main' procedure and the program.

Problem 2: 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.

Step 1: Initialize the program.
- Set the program's origin to 100h.

Step 2: Define data section.
- Declare an array 'digits' to store 7 digits.
- Define messages for user input, ascending, and descending output.

Step 3: Start of the code section.
- Begin the 'code' section.

Step 4: Define the 'main' procedure.
- Move data segment address to AX and DS.
- Call the 'initialize' procedure to get user input.
- Call the 'sort_ascending' procedure to sort digits in ascending order.
- Call the 'display_output_asc' procedure to display ascending results.
- Call the 'sort_descending' procedure to sort digits in descending order.
- Call the 'display_output_dsc' procedure to display descending results.
- Set AH to 4Ch and trigger interrupt 21h to terminate the program.

Step 5: Define the 'initialize' procedure.
- Load the address of the input message to DX and display it.
- Initialize a loop to read 7 digits from the user.
- Convert each character to a numerical value and store it in the 'digits' array.
- Increment the array index.
- Repeat the loop until 7 digits are read.

Step 6: Define the 'sort_ascending' procedure.
- Initialize variables for outer and inner loops, and a temporary variable for swapping.
- Loop through the 'digits' array and swap elements to sort in ascending order.

Step 7: Define the 'sort_descending' procedure.
- Initialize variables for outer and inner loops, and a temporary variable for swapping.
- Loop through the 'digits' array and swap elements to sort in descending order.

Step 8: Define the 'display_output_asc' procedure.
- Load the address of the ascending message to DX and display it.
- Loop through the 'digits' array and display each element with a space character.

Step 9: Define the 'display_output_dsc' procedure.
- Load the address of the descending message to DX and display it.
- Loop through the 'digits' array and display each element with a space character.

Step 10: End the 'main' procedure and the program.

4.IMPLEMENTATION

Problem 1:

```
001 org 100h
002 .stack 100h
003 .model small
004
005 .data
006     digits db 5 dup(?)
007     msg db 13, 10, "Enter the elements of array (5 digits): $"
008     avg_msg db 13, 10, "AVERAGE = $"
009     lrg_msg db 13, 10, "LARGEST = $"
010     sml_msg db 13, 10, "SMALLEST = $"
011     len db ?
012     average db ?
013     largest db ?
014     smallest db ?
015
016 .code
017
018 main proc
019     mov ax, @data
020     mov ds, ax
021
022     call initialize
023
024     call average_cal
025     call calculate
026     call display_output
027
028     mov ah, 4ch
029     int 21h
030 main endp
031
032 initialize proc
033     lea dx, msg
034     mov ah, 9
035     int 21h
036
037     mov cx, 5
038     mov si, 0
039
040 input_loop:
041     mov ah, 1
042     int 21h
043
044     sub al, '0'
045     mov digits[si], al
046
047     inc si
048
049     mov dl,32
050     mov ah,2
051     int 21h
052     loop input_loop
053
054     ret
055 initialize endp
```

```
057 average_cal proc
058     mov cx, 0000h
059     mov cl, 5
060     mov ax, 0000h
061     mov si,0
062
063 loop1:
064     add al, digits[si]
065     inc si
066     loop loop1
067
068     mov len, 5
069     div len
070     mov average, al
071
072     ret
073 average_cal endp
074
075
076 calculate proc
077     mov cx, 5
078     mov si, 0
079     mov bl, digits[si] ; Initialize largest with the first digit
080     mov bh, digits[si] ; Initialize smallest with the first digit
081     xor ax, ax
082
083 calculation_loop:
084     cmp digits[si], bl
085     jg update_largest
086
087     cmp digits[si], bh
088     jl update_smallest
089
090     jmp next_iteration
091
092 update_largest:
093     mov bl, digits[si] ; Update largest
094     jmp next_iteration
095
096 update_smallest:
097     mov bh, digits[si] ; Update smallest
098
099 next_iteration:
100     inc si
101     loop calculation_loop
102
103     mov largest, bl
104     mov smallest, bh
105
106     ret
107 calculate endp
108
```

```
display_output proc
    lea dx, avg_msg
    mov ah, 9
    int 21h

    mov dl, average
    add dl, '0'
    mov ah, 2
    int 21h

    lea dx, lrg_msg
    mov ah, 9
    int 21h

    mov dl, largest
    add dl, '0'
    mov ah, 2
    int 21h

    lea dx, sml_msg
    mov ah, 9
    int 21h

    mov dl, smallest
    add dl, '0'
    mov ah, 2
    int 21h

    ret
display_output endp

end main
```

Problem 2:

```
org 100h
.model small
.data
    digits db 7 dup(?)
    msg db "Enter the elements of array (7 digits): $"
    asc_msg db 13, 10, "Ascending: $"
    desc_msg db 13, 10, "Descending: $"

.code
main proc
    mov ax, @data
    mov ds, ax

    call initialize
    call sort_ascending
    call display_output_asc
    call sort_descending
    call display_output_dsc

    mov ah, 4ch
    int 21h
main endp
```

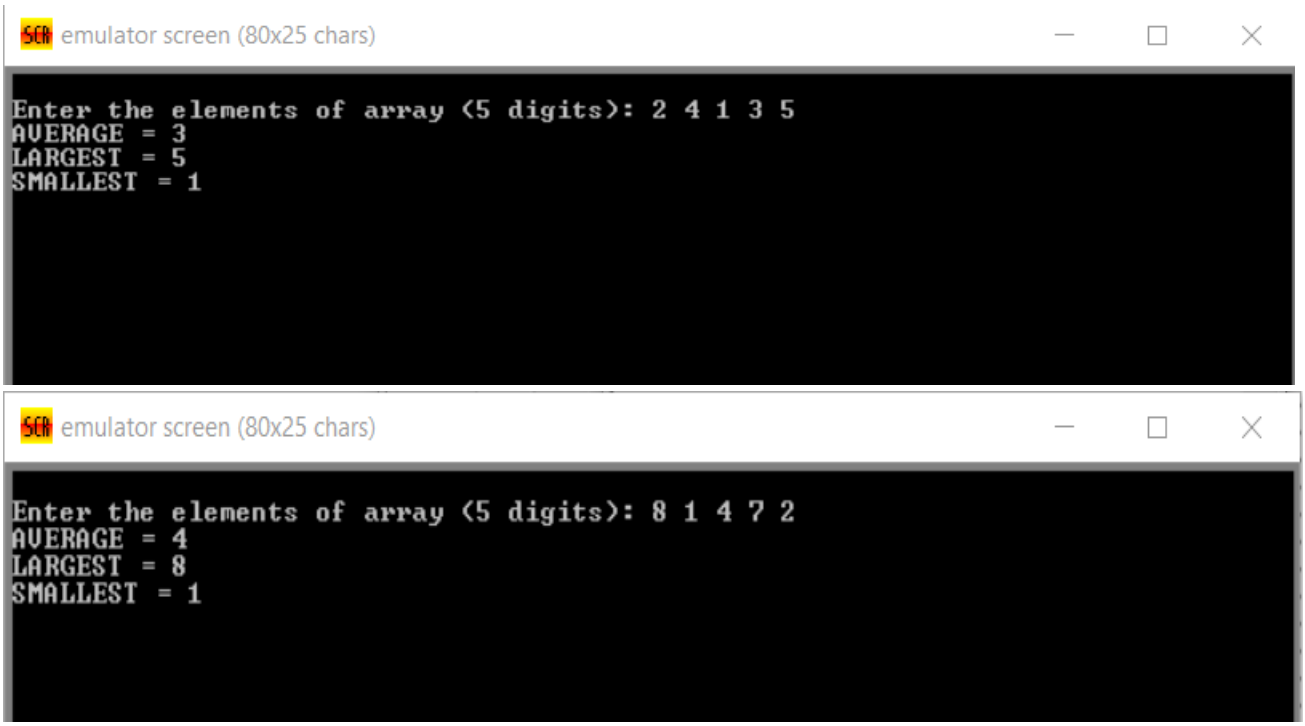
```
024 initialize proc
025     lea dx, msg
026     mov ah, 9
027     int 21h
028
029     mov cx, 7
030     mov si, 0
031
032 input_loop:
033     mov ah, 1
034     int 21h
035
036     sub al, '0'
037     mov digits[si], al
038
039     inc si
040
041     mov dl, 32
042     mov ah, 2
043     int 21h
044
045     loop input_loop
046
047     ret
048 initialize endp
049
050 sort_ascending proc
051     mov cx, 7
052     mov si, 0
053
054 asc_loop_outer:
055     mov di, si
056     inc di
057
058 asc_loop_inner:
059     mov al, digits[di]
060     cmp digits[si], al
061     jg asc_swap
062
063 asc_next_iteration:
064     inc di
065     cmp di, 7
066     jl asc_loop_inner
067     jmp asc_next_outer
068
069 asc_swap:
070     mov bl, digits[si]
071     mov digits[si], al
072     mov digits[di], bl
073     jmp asc_next_outer
074
075 asc_next_outer:
076     inc si
077     cmp si, 6
078     jl asc_loop_outer
079
080     ret
081 sort_ascending endp
082
```

```
sort_descending proc
    mov cx, 7
    mov si, 0

desc_loop_outer:
    mov di, si
    inc di

desc_loop_inner:
    mov al, digits[di]
    cmp digits[si], al
    jl desc_swap

desc_next_iteration:
    inc di
    cmp di, 7
    jl desc_loop_inner
    jmp desc_next_outer

desc_swap:
    mov bl, digits[si]
    mov digits[si], al
    mov digits[di], bl
    jmp desc_next_outer

desc_next_outer:
    inc si
    cmp si, 6
    jl desc_loop_outer

    ret
sort_descending endp

display_output_asc proc
    lea dx, asc_msg
    mov ah, 9
    int 21h

    mov cx, 7
    mov si, 0

asc_display_loop:
    mov dl, digits[si]
    add dl, '0'
    mov ah, 2
    int 21h

    mov dl, 32
    mov ah, 2
    int 21h

    inc si
    loop asc_display_loop

    ret
display_output_asc endp
```

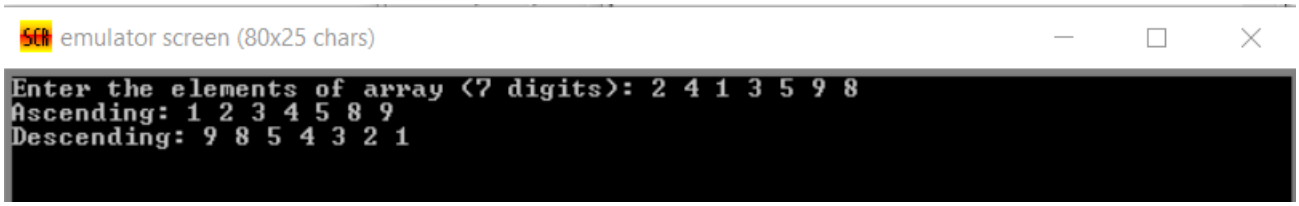
```
140 display_output_dsc proc
141     lea dx, desc_msg
142     mov ah, 9
143     int 21h
144
145     mov cx, 7
146     mov si, 0
147
148 desc_display_loop:
149     mov dl, digits[si]
150     add dl, '0'
151     mov ah, 2
152     int 21h
153
154     mov dl, 32
155     mov ah, 2
156     int 21h
157
158     inc si
159     loop desc_display_loop
160
161     ret
162 display_output_dsc endp
163
164 end main
165
```

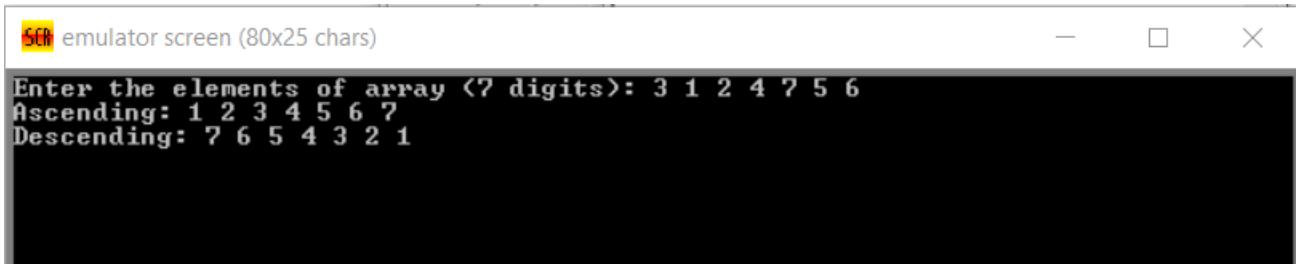
5.TEST RESULT / OUTPUT

Problem 1 Output:



Problem 2 Output:





6.ANALYSIS AND DISCUSSION

In this lab report, Problem 1 focuses on processing an array of five decimal digits, calculating the average, identifying the largest and smallest digits. The implementation likely involves iterating through the array and updating variables to keep track of the sum, largest, and smallest values. The use of procedures for each task promotes modular code organization.

In problem 2, the assembly language code takes an array of seven decimal digits and rearranges them in ascending and descending order using two distinct procedures. This task involves sorting algorithms, and the chosen approach could be bubble sort, insertion sort, or any other suitable algorithm.

7. SUMMARY:

In both problem it showcase the adaptability and control that assembly language offers in handling array operations. The use of procedures in each problem highlights the modularity of the code, facilitating easier comprehension and maintenance. Additionally, the problems underscore the low-level efficiency of assembly language in performing mathematical and logical operations, making it a suitable choice for tasks requiring precise control over hardware resources.