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: 07

Course Title: Compiler Lab

Course Code: CSE-304 Section:213-D1

Lab Experiment Name: Implement Macro in Assembly Language Programming.

Student Details

	Name	ID
1.	Irteja Mahmud	213902016

Lab Date : 30-11-2023

Submission Date : 09-12-2023

Course Teacher's Name : Sudip Ghoshal

Lab Report Status

Marks: …… **Signature:.....**

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

1. TITLE OF THE LAB EXPERIMENT

Implement Macro 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 macro.

3.PROCEDURE / ANALYSIS / DESIGN

Problem 1: Write an Assembly Language code that takes an input ARRAY and passes the array values and address to a MACRO. Now produce the summation of odd digits and even digits as output.

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 'array' to store 9 elements.
- Define messages for user prompts and result display.
- Declare variables for array size, sum, and even/odd values.

Step 3: Define macros.

- Define a 'Print_msg' macro for message printing.
- Define a 'TotalSum' macro to calculate the total sum.
- Define an 'input' macro for user input and array initialization.
- Define 'output' macros for displaying results.
- Define 'EvenDigit' and 'OddDigit' macros for displaying even and odd digits.
- Define 'SumOddEven' macro to calculate sums of odd and even digits.

Step 4: Start of the code section.

- Begin the 'code' section.

Step 5: Initialize the program.

- Move data segment address to AX and DS.
- Prompt the user to enter the array size.

- Use the 'input' macro to read and initialize the array based on user input.

Step 6: Calculate sums and display results.
- Use 'SumOddEven' macro to calculate sums of odd and even digits in the array.
- Display the results using 'OddDigit', 'EvenDigit', and 'Total_Sum' macros.

Step 7: End the program.
- Set AH to 4Ch and trigger interrupt 21h to terminate the program.

4.IMPLEMENTATION

Problem 1:

```
org 100h
.model small
.stack 100h
.data
    array db 9 dup(?)
    msgInput db "ENTER THE SIZE OF ARRAY : $"
    Inputmsg db 10,13,"ENTER THE ARRAY : $"
    msgOdd db 10,13, "ODD DIGITS : $"
    msgEven db 10,13, "EVEN DIGITS : $"
    msgTotal db 10,13, "TOTAL SUM : $"
    sum dw 0
    k dw ?
    EvenVal dw 0
    OddVal dw 0

Print_msg macro m
     mov ah,9
     lea dx,m
     int 21h
endm

TotalSum macro a

    mov bh, a
    mov al, bh
    add al, 1
    mul bh
    mov dl, 2
    div dl
    mov sum, ax

endm

```

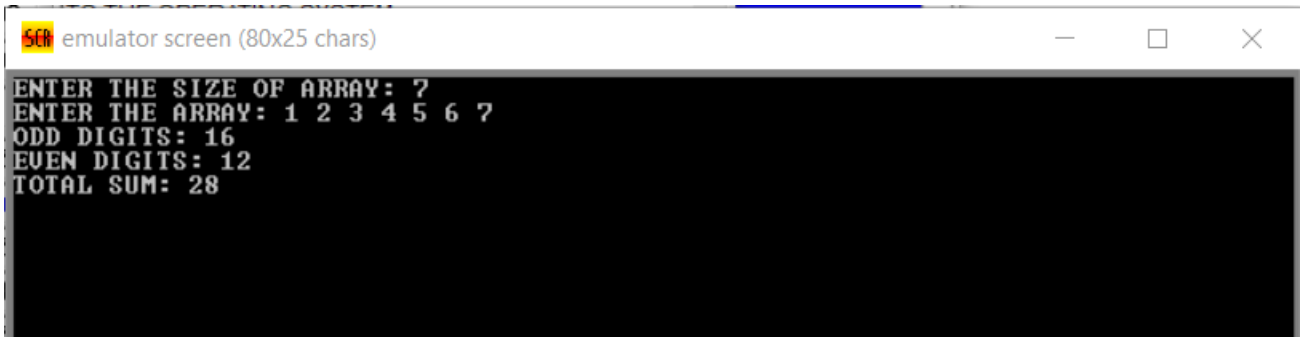
```
input macro
    mov ah, 1
    int 21h
    sub al, '0'

    mov bl,al

    TotalSum al
    mov bh,0
    mov k,bx

    mov cx, k
    lea di, array

    Print_msg Inputmsg
inputLoop:

    mov ah,1
    int 21h
    sub al,48

    mov [di],al

    mov ah,2
    mov dl,32
    int 21h

    inc di
    loop inputLoop
endm

output macro value
     mov ah, 2
     mov dl,value
     add dl,48
     int 21h
endm

EvenDigit macro
   Print_msg msgEven
   mov ax,EvenVal
   mov bl,10
   div bl
   mov bx,ax
   output bl
   output bh
endm

OddDigit macro
   Print_msg msgOdd
   mov ax,OddVal
   mov bl,10
   div bl

   mov bx,ax
   output bl
   output bh
endm
```

```
Total_Sum macro
   Print_msg msgTotal
   mov ax,sum
   mov bl,10
   div bl

   mov bx,ax
   output bl
   output bh
endm

SumOddEven macro
    mov cx,k
    lea di,array
calculateLoop:
    mov al, [di]


    and [di],1
    jnz OddSum

    mov ah,0
    add EvenVal,ax
    jmp nextDigit

OddSum:
    mov ah,0
    add OddVal,ax
    jmp nextDigit

nextDigit:
    inc di
    loop calculateLoop

endm


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

    Print_msg msgInput
    input

    SumOddEven
    OddDigit
    EvenDigit
    Total_Sum


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

5.TEST RESULT / OUTPUT

Problem 1 Output:

Test Case 1:



Test Case 2:

6.ANALYSIS AND DISCUSSION

In this lab report, that problem exhibits a well-structured and modular design, leveraging macros to encapsulate specific functionalities and promote code reusability. The program efficiently handles user input, prompting for the array size and elements, and utilizes optimized arithmetic operations for total sum calculation. The macros dedicated to processing odd and even digits, along with those responsible for displaying results, contribute to code readability and maintainability. Registers are effectively utilized throughout the code, showcasing prudent resource management. Overall, the program successfully achieves its objective of processing an array of digits, calculating sums based on digit parity, and presenting clear results to the user, demonstrating a thoughtful and organized approach to assembly programming.

7. SUMMARY:

This 8086-assembly code demonstrates effective programming practices, including modular code design through macros, proper user input handling, efficient arithmetic operations, and clear display of results. The program successfully achieves its goal of processing an array of digits, categorizing them into odd and even, calculating their sums, and presenting the results to the user. The use of macros enhances code readability and reusability, contributing to a well-organized and structured program.