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: 05

Course Title: Compiler Lab

Course Code: CSE-304 Section:213-D1

Lab Experiment Name: Implement Array and String in Assembly Language Programming.

Student Details

	Name	**ID**
1.	Irteja Mahmud	213902016

Lab Date : 02-11-2023

Submission Date : 08-12-2023

Course Teacher's Name : Sudip Ghoshal

Lab Report Status

Marks: …………………………………… Signature:.....................

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

1. TITLE OF THE LAB EXPERIMENT

Implement Array and String in Assembly Language Programming.

2. OBJECTIVES/AIM

To understand the use of Array & String in Assembly Language Program.

3.PROCEDURE / ANALYSIS / DESIGN

Problem 1: Write an assembly language code to take natural number series as input and as output, show:
a. The summation of odd numbers.
b. The summation of even numbers.

Step 1: Initialize the program.
- 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.
- Define variables for array size, even sum, and odd sum.

Step 3: Define macros.
- Define a 'print_msg' macro for message printing.
- Define an 'input' macro for user input and array initialization.
- Define an 'output' macro for displaying results.

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.

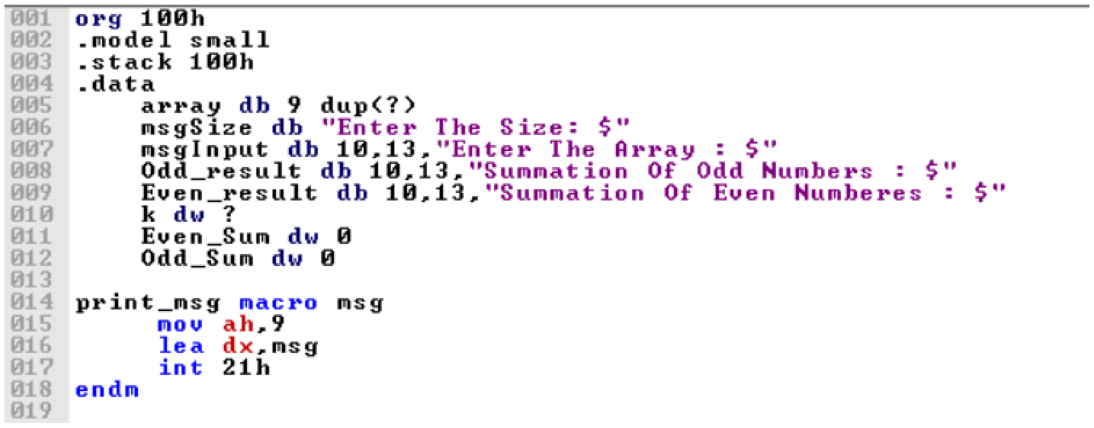
Step 6: Calculate the sum of even and odd numbers.
- Use a loop to iterate through the array elements.
- Check if the current digit is even or odd using bitwise AND.
- Accumulate the sum of even and odd numbers separately.

Step 7: Display the results.
- Use the 'print_msg' macro to display messages.
- Use the 'output' macro to display the sum of odd and even numbers.

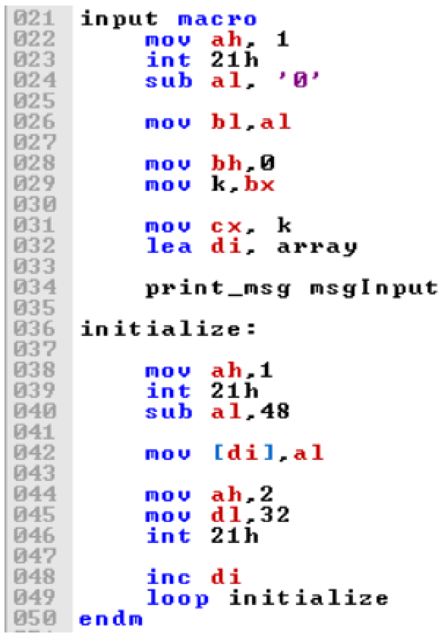
Step 8: End the program.
- Set AH to 4Ch and trigger interrupt 21h to terminate the program.

4.IMPLEMENTATION

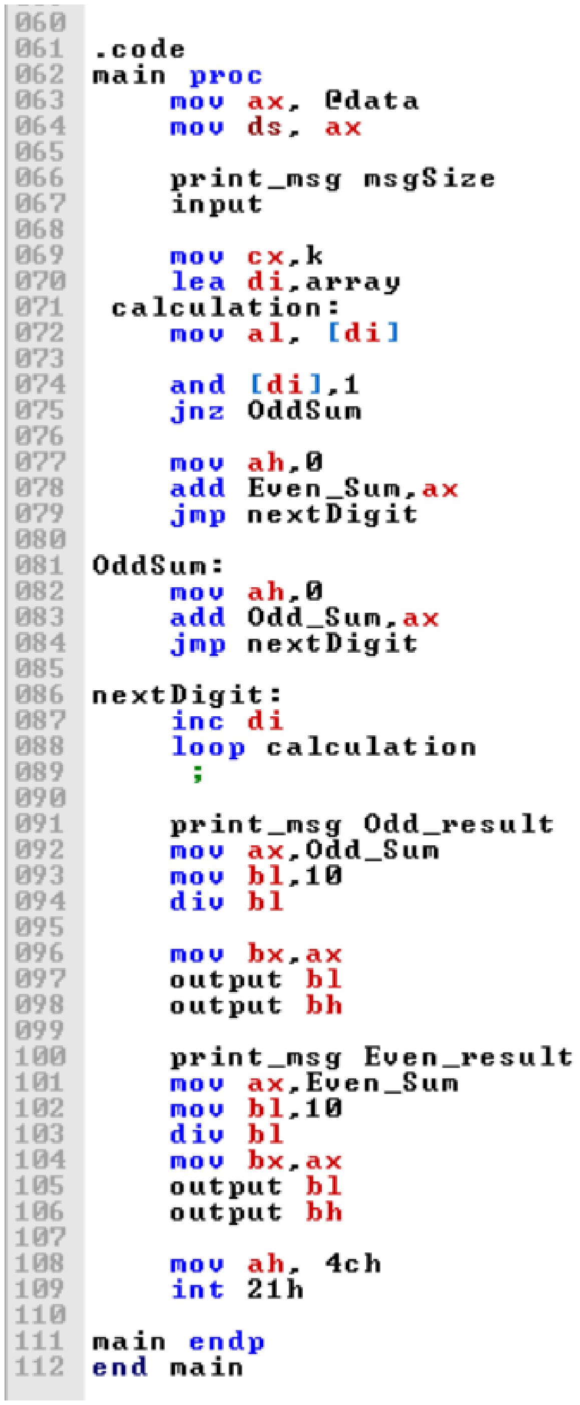
Problem 1:



```
org 100h
.model small
.stack 100h
.data
        array db 9 dup(?)
        msgSize db "Enter The Size: $"
        msgInput db 10,13,"Enter The Array : $"
        Odd_result db 10,13,"Summation Of Odd Numbers : $"
        Even_result db 10,13,"Summation Of Even Numberes : $"
        k dw ?
        Even_Sum dw 0
        Odd_Sum dw 0

print_msg macro msg
          mov ah,9
          lea dx,msg
          int 21h
endm

```



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

        mov bl,al

        mov bh,0
        mov k,bx

        mov cx, k
        lea di, array

        print_msg msgInput

initialize:

        mov ah,1
        int 21h
        sub al,48

        mov [di],al

        mov ah,2
        mov dl,32
        int 21h

        inc di
        loop initialize
endm
```

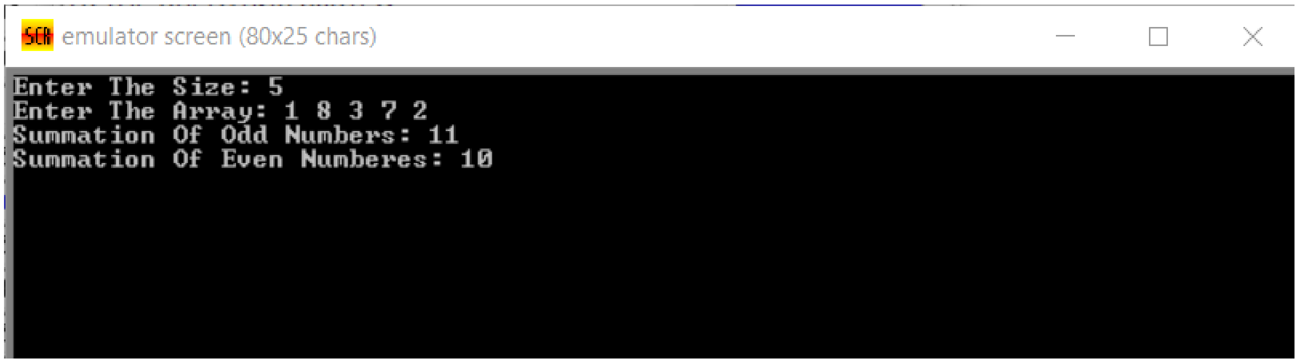


```
.code
main proc
    mov ax, @data
    mov ds, ax

    print_msg msgSize
    input

    mov cx,k
    lea di,array
 calculation:
    mov al, [di]

    and [di],1
    jnz OddSum

    mov ah,0
    add Even_Sum,ax
    jmp nextDigit

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

nextDigit:
    inc di
    loop calculation
     ;

    print_msg Odd_result
    mov ax,Odd_Sum
    mov bl,10
    div bl

    mov bx,ax
    output bl
    output bh

    print_msg Even_result
    mov ax,Even_Sum
    mov bl,10
    div bl
    mov bx,ax
    output bl
    output bh

    mov ah, 4ch
    int 21h

main endp
end main
```

5.TEST RESULT / OUTPUT

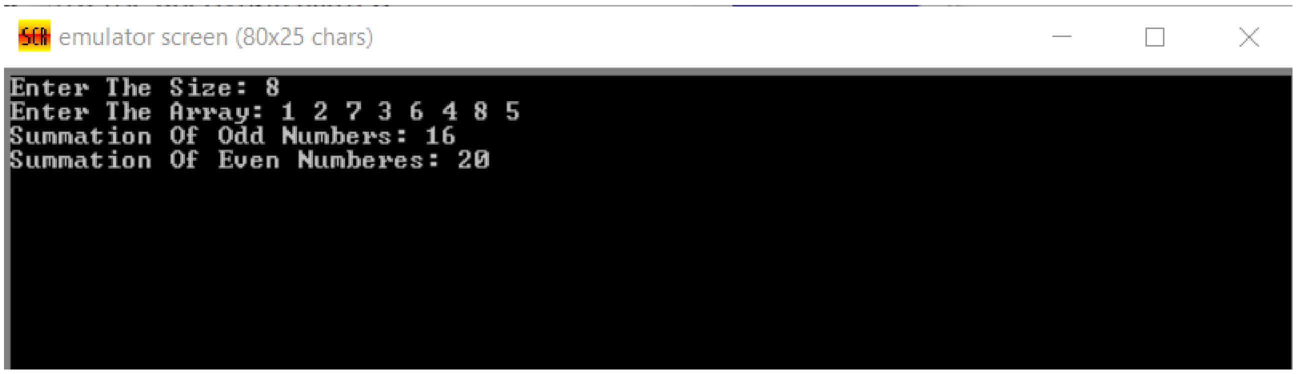
Problem 1 Output:

Test Case 1:



```
Enter The Size: 5
Enter The Array: 1 8 3 7 2
Summation Of Odd Numbers: 11
Summation Of Even Numberes: 10
```

Test Case 2:



```
Enter The Size: 8
Enter The Array: 1 2 7 3 6 4 8 5
Summation Of Odd Numbers: 16
Summation Of Even Numberes: 20
```

6.ANALYSIS AND DISCUSSION

In this lab report, the program prompts the user to input the size of an array and initializes the array accordingly. Utilizing macros for code modularity, the algorithm then calculates the sum of even and odd numbers within the array, distinguishing between them based on bitwise AND operations. The results are displayed, showcasing the sum of odd and even numbers separately. The code structure is clear and organized, leveraging macros for improved readability. The program demonstrates effective use of conditional statements within a loop for array processing and arithmetic operations. Overall, it provides a practical example of user-driven input handling, array manipulation, and conditional processing in assembly language.

7. SUMMARY:

In conclusion, these codes contribute to a foundational understanding of loop-based programming in the context of array operations and user input handling. They provide practical examples of how loops are employed in real-world scenarios within assembly language programming.