Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Fall, Year: 2025), B.Sc. in CSE (Day)

Lab Report NO: 04
Course Title: Microprocessors, Microcontrollers, and Embedded System Lab
Course Code: CSE 304
Section: 232-D1

Lab Experiment Name: Implementation of loop using assembly language

Student Details

	Name	ID
1.	Ashab Uddin	232002274

Lab Date : **29/10/2025**
Submission Date : **05/11/2025**
Course Teacher's Name : **Jarin Tasnim Tonvi**

Lab Report Status

Marks: …… **Signature: .....**
Comments: ..... **Date: .....**

1. INTRODUCTION

This lab showed how to use loops in assembly language to work with arrays and repeat tasks using counters and jump commands. In high-level languages, we use loops like for and while, but in assembly, loops are made with labels, jump instructions, and working with registers.

In this experiment, we learned how to get several inputs, save them in an array, and use bitwise operations to find out if each number is even or odd. We went through the array step by step with a loop and changed counters based on the results. This task helped us see how basic decisions and going through lists happen in low-level programming.

2. OBJECTIVES

- To understand how arrays are used in assembly language.
- To learn how loops are created using jump instructions.
- To apply bitwise operations for decision-making.
- To count and display even and odd numbers using assembly logic.
-

3. PROCEDURE

Problem: Count Even and Odd Numbers from an Array

Here's a simplified explanation of the problem and steps for counting even and odd numbers from an array in assembly language:

1. The program asks the user how many numbers they want to enter (from 1 to 20).
2. The user enters a number as an ASCII code, so the program converts it to a real number by subtracting 30H.
3. Then, a loop runs to read each number and save it into an array called ARRAY.
4. After saving all the numbers, another loop checks each number one by one.
5. The program uses a bitwise AND with 1 (AL AND 1) to test the number:
    - If the result is 0, the number is even.
    - If the result is 1, the number is odd.
6. It keeps two counters: one for even numbers and one for odd numbers.
7. At the end, the program shows the total count of even and odd numbers on the screen using the INT 21H function.

4. IMPLEMENTATION

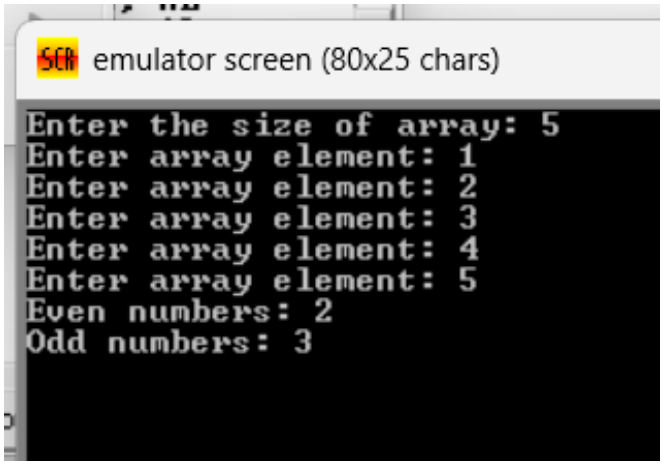
CODE

```
.MODEL SMALL
.STACK 100H

.DATA
```

```
    ARRAY DB 20 DUP(?)
    N DB ?
    EVEN_COUNT DB 0
    ODD_COUNT DB 0

    STR1 DB 'Enter the size of array: $'
    STR2 DB 0DH,0AH,'Enter array element: $'
    STR3 DB 0DH,0AH,'Even numbers: $'
    STR4 DB 0DH,0AH,'Odd numbers: $'

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX



    LEA DX, STR1
    MOV AH, 9
    INT 21H

    MOV AH, 1
    INT 21H
    SUB AL, 30H
    MOV N, AL

    MOV EVEN_COUNT, 0
    MOV ODD_COUNT, 0



    XOR SI, SI

INPUT_LOOP:
    MOV BL, N
    CMP SI, BX
    JAE COUNT_LOOP

    LEA DX, STR2
    MOV AH, 9
    INT 21H

    MOV AH, 1
    INT 21H
    SUB AL, 30H
    MOV ARRAY[SI], AL

    INC SI
    JMP INPUT_LOOP


; Count even and odd

COUNT_LOOP:
```

```
    XOR SI, SI

COUNT_NEXT:
    MOV BL, N
    CMP SI, BX
    JAE DISPLAY_RESULT

    MOV AL, ARRAY[SI]
    AND AL, 1
    CMP AL, 0
    JE IS_EVEN

    INC ODD_COUNT
    JMP NEXT_INDEX

IS_EVEN:
    INC EVEN_COUNT

NEXT_INDEX:
    INC SI
    JMP COUNT_NEXT



; Display results

DISPLAY_RESULT:
    LEA DX, STR3
    MOV AH, 9
    INT 21H

    MOV DL, EVEN_COUNT
    ADD DL, 30H
    MOV AH, 2
    INT 21H

    LEA DX, STR4
    MOV AH, 9
    INT 21H

    MOV DL, ODD_COUNT
    ADD DL, 30H
    MOV AH, 2
    INT 21H

    MOV AH, 4CH
    INT 21H

MAIN ENDP
END MAIN
```

5. OUTPUT



6. ANALYSIS AND DISCUSSION

In this lab, we used loops, arrays, and bit operations in assembly language. The program first asks for the array size, then runs a loop to save each number in the array. Next, another loop checks if every number is even or odd.

The AND AL, 1 command was important because it looks at the last bit of the number. If Last bit = 0 means even and If Last bit = 1 means odd

This bit trick is faster than dividing numbers. The lab showed how loops use jump commands inside, how arrays work in assembly, and how binary checks make quick choices. It helped understand basic data storage and repeating tasks at a low level.