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
Course Code: CSE 304 **Section:** 232-D1

Lab Experiment Name: Print number of even numbers and odd numbers in the array(even count & odd count).

Student Details

Name		ID
1.	Rukonuzzaman Topu	232002280

Submission Date : 03/12/2025

Course Teacher's Name : Jarin Tasnim Tonvi

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

Lab Report Status

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

1. INTRODUCTION

This lab demonstrated how loops work in assembly language when handling arrays and repeating operations. In higher-level languages, we rely on for or while loops, but in assembly we create loops using labels, jump instructions, and register values. During the experiment, we practiced taking multiple inputs, storing them in an array, and using bitwise checks to determine whether each number was even or odd. We processed the array element by element using a loop and updated counters based on the results. Overall, this experiment helped us understand how simple decision-making and iteration are performed in low-level programming.

2. OBJECTIVES

1. To gain an understanding of how arrays are managed and accessed in assembly language.
2. To learn the process of creating loops using labels and jump instructions.
3. To use bitwise operations to make logical decisions in the program.
4. To implement counting and displaying of even and odd numbers using assembly-level techniques.

3. PROCEDURE

This program demonstrates how to count even and odd numbers from an array using assembly language.

1. First, the program asks the user how many values they want to input (between 1 and 20).
2. Since the user's input comes in as an ASCII character, the program converts it to a numeric value by subtracting 30H.
3. A loop then runs to collect each number from the user and store it in an array named ARRAY.
4. Once all the numbers are stored, another loop goes through the array one element at a time.
5. To determine if a number is even or odd, the program performs a bitwise AND operation with 1:
    1) If the result is 0, the number is even.
    2) If the result is 1, the number is odd.
6. Two separate counters are used to keep track of how many even and odd values were found.
7. Finally, the program displays the total number of even and odd entries on the screen using the INT 21H output function.

4. IMPLEMENTATION

Source 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
```

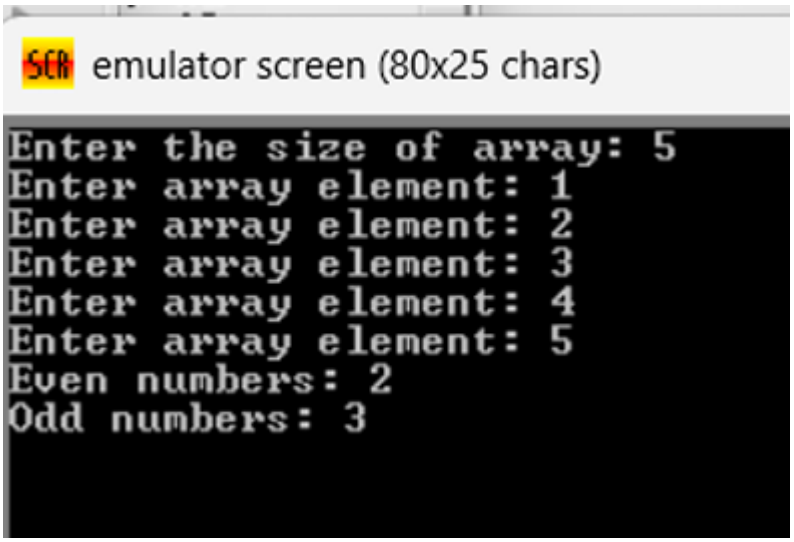
```
    LEA DX, STR4
    MOV AH, 9
    INT 21H

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

    MOV AH, 4CH
    INT 21H

MAIN ENDP
END MAIN
```

Output :



DISCUSSION

In this lab, we worked with loops, arrays, and bitwise operations in assembly language. The program begins by asking the user for the size of the array, and then a loop is used to read each number and store it in memory. After the inputs are collected, another loop goes through all the stored values to determine whether each one is even or odd. The instruction **AND AL, 1** is crucial because it checks the least significant bit of the number—if the last bit is **0**, the number is even, and if it is **1**, the number is odd. This method is much faster than performing division to check parity. Overall, the experiment demonstrated how loops rely on jump instructions, how arrays are handled in assembly, and how simple binary checks allow quick decisions. It provided a clearer understanding of data handling and repetitive tasks at a

low-level programming stage.logic is implemented at the machine level, enhancing the ability to write more complex assembly programs in the future.