



**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:** Microprocessor & Microcontroller Lab

**Course Code:** CSE 304

**Section:** 213D1

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

**Student Details**

	<b>Name</b>	<b>ID</b>
<b>1.</b>	Md Javed Hossen	213902046

**Lab Date** : 20-10-2023  
**Submission Date** : 21-12-2023  
**Course Teacher's Name** : Sudip Ghoshal

**Lab Report Status**

**Marks:** .....

**Signature:**.....

**Comments:**.....

**Date:**.....

## **1. TITLE OF THE LAB REPORT EXPERIMENT**

Implement Array and String in Assembly Language Programming

## **2. OBJECTIVES/AIM**

The objectives of this exercise are to gain a practical understanding of arrays and strings in Assembly Language, familiarize with the 'dup' operator for initializing arrays and creating strings, and comprehend the use of pointer registers such as SI, DI, BX, and BP within the context of program development

## **3. PROCEDURE**

**Problem: Sum of even and odd number from natural number series 1+2+3...+n**

Step 1: Start

Step 2: Initialize data segment arr of 10 size, I, temp to 0 and oddSum,evenSum to empty

Step 3: Include data segment into main procedure into 'DS' register Step 4: take array size from user and copy to 'bl' and 'cl' register to set 'cx' value

Step 5: set pointer or index register to the 'arr' using 'si'.

Step 6: Continue loop. That will continue cx=cl to 0.

'taken\_input' Step 7: Take input and store array [si], al.

Increment 'si' pointer by 1 'inc si'. Print a space between each input Continue 'loop taken\_input' and repeat step-7

Step 8: Again point array using 'si' for loop to calculate even and odd number summation

Step 9: set 'dl' as 2 to divide each element of the array to check even or odd number

Step 10: take 'arr\_traverse' level to jumping conditionally again and again Initially set 'ah' as 0.

Take element from array using [si] into 'al'.copy 'al' to 'temp' variable for future Now divide 'al' by 'dl' note: ah=remainder and al= quotient Compare 'ah' with 0 or not. If 0 this is even

number else  
 odd number Jump 'JE' even\_sum :jump equal to 0  
 Jump 'JNE' odd\_sum jump not equal to 0  
 After jumping we need to get back into 'arr\_traverse' loop. So take  
 a  
 level 'back' After backing, increment variable 'i' by 1 and  
 'inc si' Compare 'i' with 'bl'. If equal called 'print\_result'  
 level  
 If less then again called 'arr\_traverse' level  
 Step 11: Continue Step-10  
 Step 12: Print even and odd summation into console.  
 Step 13: End

#### 4. IMPLEMENTATION

**Problem: Sum of even and odd number from natural number series 1+2+3...+n**

```

.model small
.stack 100h
; Macro for printing a string
printString macro str
    lea dx, str
    mov ah, 09h
    int 21h
endm
; Macro for printing a character in DL
printChar macro
    mov ah, 02h
    int 21h
endm

```

**.data**

arr db 100 dup(?)

evenSum db 0

i db 0

oddSum db 0

newline db 10, 13, '\$'

msgSize db 'Enter size of the array: \$' msgElement db

'Enter array element: \$' msgEvenSum db 'Even summation is: \$'

msgOddSum db 'Odd summation is: \$'

**.code**

main proc

; Initialize data segment

mov ax, @data

mov ds, ax

; Take array size from user

printString msgSize

mov ah, 1

int 21h

sub al, 48

mov bl, al

mov i,0

; Take array elements from user

mov si, offset arr

printString newline

printString msgElement

take\_input:

mov ah, 1

int 21h

sub al, 48

mov [si], al

inc si

inc i

```

    cmp i, bl
    JL take_input
    ; Traverse array and calculate sum of odd and even numbers
mov si, offset arr
    mov dl, 2
    mov i, 0
arr_traverse:
    mov al, [si]
    div dl
    cmp ah, 0
    je even_sum
    jmp odd_sum
even_sum:
    mov al, evenSum
    add al, [si]
    mov evenSum, al
    jmp back
odd_sum:
    mov al, oddSum
    add al, [si]
    mov oddSum, al
    jmp back
back:
    inc si
    inc i
    cmp i, bl
    JL arr_traverse
    ; Print the results
print_result:
    printString newline
    printString msgEvenSum mov dl, evenSum
    add dl, 48

```

```

    printChar
    printString newline
    printString msgOddSum mov dl, oddSum
    add dl, 48
    printChar
    ; Exit the program
    mov ah, 4ch
    int 21h
main endp
end main

```

## 5. OUTPUT

The image shows a screenshot of an x86-64 emulator window titled "emulator screen (80x27 chars)". The window is split into two panes. The left pane displays assembly code, and the right pane shows the program's output.

**Assembly Code (Left Pane):**

```

mov evenSum, 0
jmp back
odd_sum:
mov al, oddSum
add al, [si]
mov oddSum, al
jmp back
back:
inc si
inc i
cmp i, bl
jl arr_traverse

; Print the results
print_result:
printString newline
printString msgEvenSum
mov dl, evenSum
add dl, 48
printChar

printString newline
printString msgOddSum
mov dl, oddSum
add dl, 48
printChar

; Exit the program
mov ah, 4ch

```

**Output (Right Pane):**

```

Enter size of the array: 5
Enter array element: 12345
Even summation is: 6
Odd summation is: 9

```

Figure-1: Summation of even and odd from 5th natural number series using array

## **6. ANALYSIS AND DISCUSSION**

I've successfully developed a program that calculates the sum of even and odd numbers from a user-input natural number series and stores the results in an array. The program prompts the user for the number of terms in the series and then uses a loop to collect the numbers. During testing, I entered the fifth term of a series and the program correctly displayed the sums in the console. However, it's important to note that if we input a series with more than six terms, the console may show special characters because ASCII values are limited to representing the digits 0 to 9. Nevertheless, the program can accurately store and handle sums for any number of terms within a variable.