Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (fall, Year:2024), B.Sc. in CSE (Day)

Lab Report #04

Course title: Microprocessor & Microcontroller Lab

Course Code: CSE 304

Section: 222_D13

Lab Experiment Name: Implementation of loop using assembly language**.**

Student Details

Name	ID
Md. Moshiur Rahman	221902324

Submission Date : 23/11/2024

Course Teacher's Name : Md. Jahid Tanvir

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

Lab Report Status

Marks: …… **Signature:.....**

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

1. TITLE OF THE LAB REPORT EXPERIMENT

Implementation of loop using assembly language.

2. OBJECTIVES

• To gather knowledge how to use loop in assembly language.

Take numbers as input from the user and print whether the given number is odd or even. You have to iterate the process until user press "N". If user press "N" terminate your program otherwise for given number print whether it is odd or even.

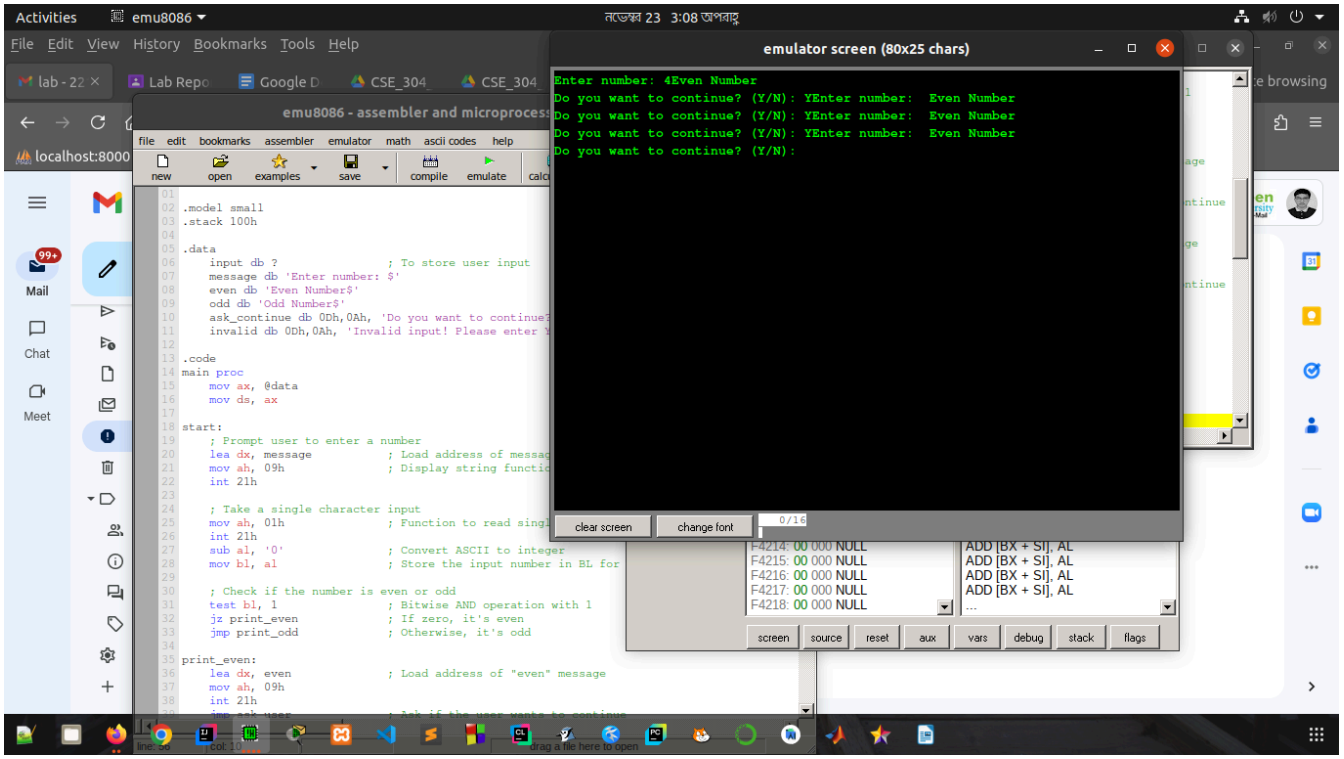
3. IMPLEMENTATION

Src code:

```
.model small
.stack 100h

.data
    input db ?              ; To store user input
    message db 'Enter number: $'
    even db 'Even Number$'
    odd db 'Odd Number$'
    ask_continue db 0Dh,0Ah, 'Do you want to continue? (Y/N): $'  ; Prompt message
    invalid db 0Dh,0Ah, 'Invalid input! Please enter Y or N$'

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

start:
    ; Prompt user to enter a number
    lea dx, message         ; Load address of message
    mov ah, 09h             ; Display string function
    int 21h

    ; Take a single character input
    mov ah, 01h             ; Function to read single character
    int 21h
    sub al, '0'         ; Convert ASCII to integer
    mov bl, al              ; Store the input number in BL for further use

    ; Check if the number is even or odd
    test bl, 1          ; Bitwise AND operation with 1
    jz print_even           ; If zero, it's even
    jmp print_odd           ; Otherwise, it's odd

print_even:
    lea dx, even            ; Load address of "even" message
    mov ah, 09h
    int 21h
    jmp ask_user            ; Ask if the user wants to continue

print_odd:
```

```
    lea dx, odd             ; Load address of "odd" message
    mov ah, 09h
    int 21h
    jmp ask_user            ; Ask if the user wants to continue

ask_user:
    ; Prompt the user to continue or exit
    lea dx, ask_continue
    mov ah, 09h
    int 21h

    ; Take user input (Y/N)
    mov ah, 01h
    int 21h
    cmp al, 'N'             ; Check if user pressed 'N'
    je exit             ; Exit if 'N'
    cmp al, 'n'             ; Also check for lowercase 'n'
    je exit
    cmp al, 'Y'             ; Check if user pressed 'Y'
    je start            ; Repeat the process if 'Y'
    cmp al, 'y'             ; Also check for lowercase 'y'
    je start

    ; Handle invalid input
    lea dx, invalid
    mov ah, 09h
    int 21h
    jmp ask_user            ; Prompt again

exit:
    mov ah, 4Ch             ; Terminate program
    int 21h

main endp
end main
```

Test(Output)



Use a loop to find out the summation of 1+3+5+7+.....+99. Also try to find out the summation using formula.

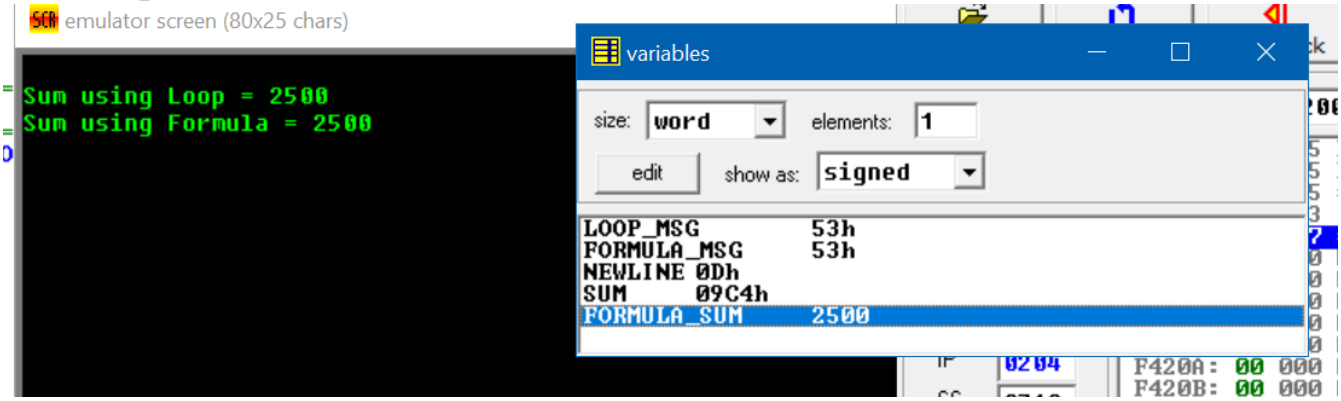
Src code:

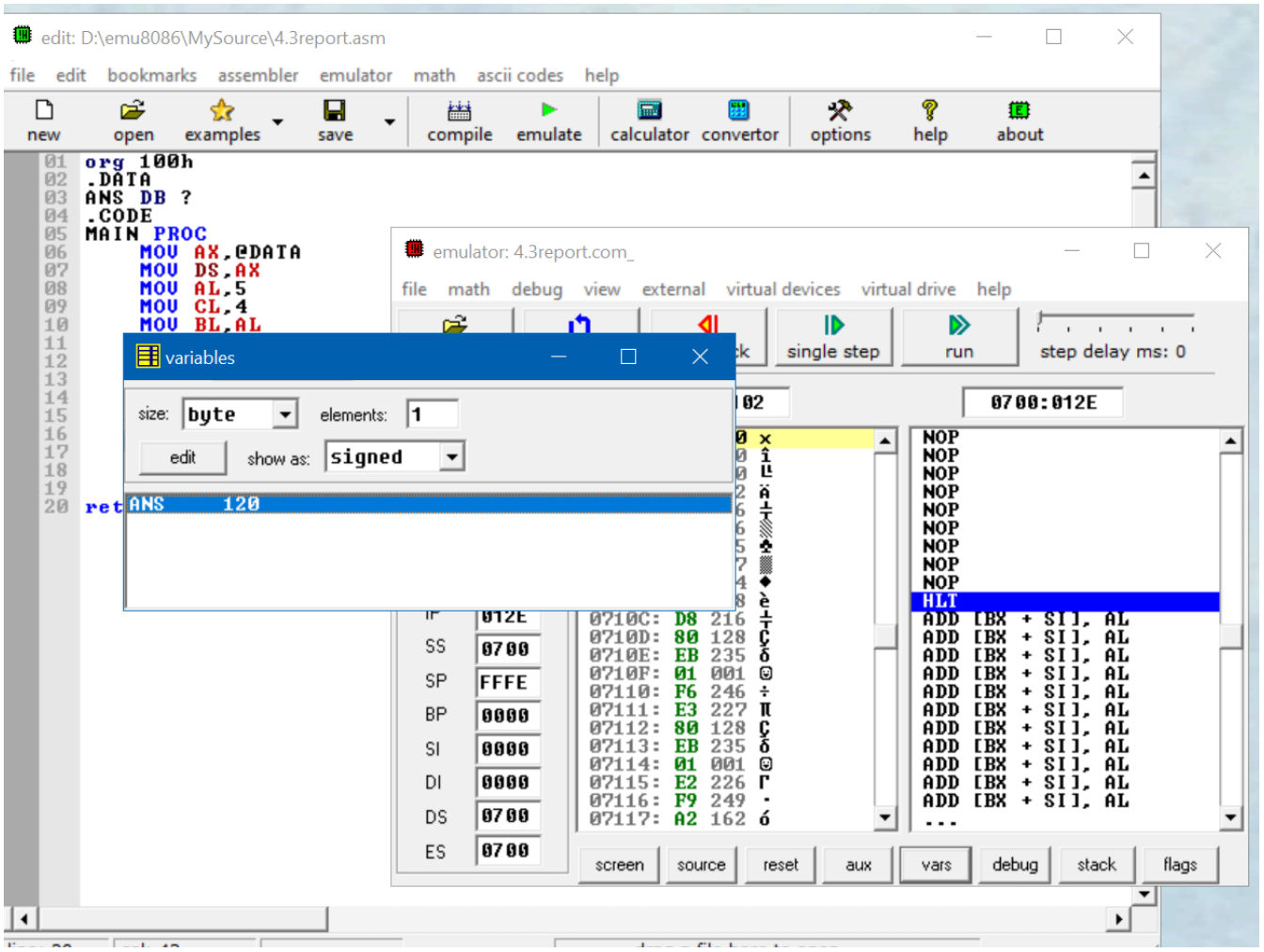
```
.MODEL SMALL
.STACK 100H
.DATA
        loop_msg db "Sum using Loop = $"
        formula_msg db "Sum using Formula = $"
        newline db 0DH, 0AH, '$'
        sum dw 0          ; Store the sum using loop
        formula_sum dw 2500 ; Pre-calculated sum using formula (50^2)

.CODE
MAIN PROC
        ; Initialize data segment
        MOV AX, @DATA
        MOV DS, AX

        ; Display newline
        LEA DX, newline
        MOV AH, 09H
        INT 21H

        ; ==========================
        ; Summation using Loop
        ; ==========================
        MOV CX, 1       ; CX = 1 (starting odd number)
        MOV BX, 99              ; BX = 99 (limit)
        MOV AX, 0       ; AX = 0 (to store sum)

sum_loop:
```

```
        ADD AX, CX              ; Add current odd number to sum
        ADD CX, 2       ; Move to next odd number
        CMP CX, BX              ; Check if reached limit
        JG end_loop     ; Exit if CX > BX
        JMP sum_loop            ; Repeat loop

end_loop:
        MOV sum, AX             ; Store the final sum in `sum`

        ; Display message for sum using loop
        LEA DX, loop_msg
        MOV AH, 09H
        INT 21H

        ; Display result (sum)
        MOV AX, sum
        CALL print_number

        ; ==============================
        ; Display pre-calculated sum using formula
        ; ==============================
        LEA DX, newline
        MOV AH, 09H
        INT 21H

        LEA DX, formula_msg
        MOV AH, 09H
        INT 21H

        ; Display result (50^2)
        MOV AX, formula_sum
        CALL print_number

        ; Exit program
        MOV AH, 4CH
        INT 21H
MAIN ENDP

; ================================
; Subroutine to print a number in AX
; ================================
print_number PROC
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX

        XOR CX, CX              ; Digit count = 0
        MOV BX, 10              ; Base 10 for decimal conversion
number_loop:
        XOR DX, DX              ; Clear DX before division
        DIV BX          ; AX / 10, remainder in DX, quotient in AX
        PUSH DX         ; Store remainder (digit)
        INC CX          ; Increment digit count
        CMP AX, 0       ; Check if quotient is 0
        JNE number_loop         ; Repeat if not 0

        ; Print digits
print_digits:
```

```
        POP DX          ; Get digit from stack
        ADD DL, '0'     ; Convert to ASCII
        MOV AH, 02H             ; Print character function
        INT 21H
        LOOP print_digits  ; Loop until all digits are printed

        POP DX
        POP CX
        POP BX
        POP AX
        RET
print_number ENDP

END MAIN
```

Test(Output)



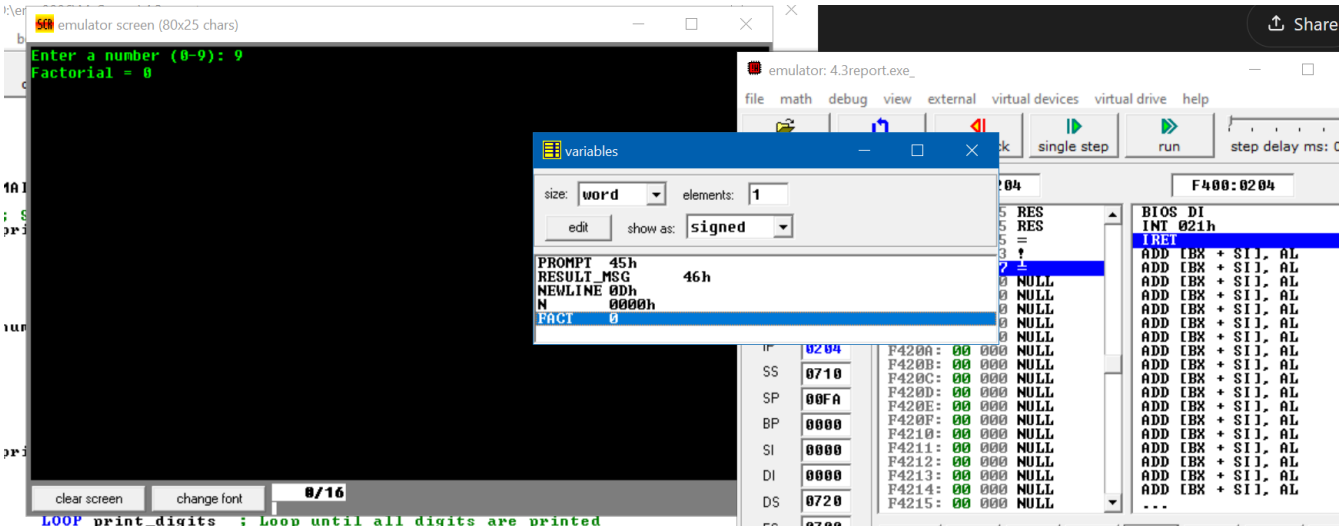


; Program to calculate factorial of a number in 8086 assembly

```
; Works for small values of n (result < 16-bit limit)
.MODEL SMALL
.STACK 100H
.DATA
        prompt db "Enter a number (0-9): $"
        result_msg db "Factorial = $"
        newline db 0DH, 0AH, '$'
        n dw 0
        fact dw 1

.CODE
MAIN PROC
        ; Initialize data segment
        MOV AX, @DATA
        MOV DS, AX

        ; Display prompt
        LEA DX, prompt
        MOV AH, 09H
        INT 21H

        ; Read input character
        MOV AH, 01H
        INT 21H
        SUB AL, '0'     ; Convert ASCII to integer
        MOV CX, AX      ; Store number in CX

        ; Calculate factorial
        MOV AX, 1       ; AX = 1 (initial factorial value)
```

```
        MOV BX, 1       ; BX = counter
factorial_loop:
        CMP BX, CX      ; Check if counter reached n
        JA done         ; Exit loop if BX > n
        MUL BX          ; Multiply AX by BX (AX = AX * BX)
        INC BX          ; Increment counter
        JMP factorial_loop

done:
        ; Store result
        MOV fact, AX

        ; Display newline
        LEA DX, newline
        MOV AH, 09H
        INT 21H

        ; Display result message
        LEA DX, result_msg
        MOV AH, 09H
        INT 21H

        ; Convert factorial to string
        MOV AX, fact
        CALL print_number ; Print the factorial result

        ; Exit program
        MOV AH, 4CH
        INT 21H
MAIN ENDP

; Subroutine to print a number in AX
print_number PROC
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX

        XOR CX, CX      ; Digit count = 0
number_loop:
        XOR DX, DX      ; Clear DX before division
        DIV BX          ; AX / 10, remainder in DX, quotient in AX
        PUSH DX         ; Store remainder (digit)
        INC CX          ; Increment digit count
        CMP AX, 0       ; Check if quotient is 0
        JNE number_loop         ; Repeat if not 0

        ; Print digits
print_digits:
        POP DX          ; Get digit from stack
        ADD DL, '0'     ; Convert to ASCII
        MOV AH, 02H             ; Print character function
        INT 21H
        LOOP print_digits  ; Loop until all digits are printed

        POP DX
        POP CX
        POP BX
        POP AX
        RET
print_number ENDP

END MAIN
```

TEST/OUTPUT



4. ANALYSIS AND DISCUSSION [3 marks]

Based on the focused objective(s) to understand about the loops in assembly language and the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

5. SUMMARY:

Summation: Use a loop to sum odd numbers (1 to 99). Calculate using the formula n2n2 (e.g., 502=2500502=2500).

Factorial: Input nn; calculate factorial iteratively (n!=1×2×⋯×nn!=1×2×⋯×n).

Odd/Even Check: Continuously input numbers, print odd/even until the user presses "N" to terminate the program.