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: 03
Course Title: Microprocessors, Microcontrollers, and Embedded System Lab
Course Code: CSE 304
Section: 232-D1

Lab Experiment Name: Implementation of conditional statement using assembly language

Student Details

	Name	ID
1.	Ashab Uddin	232002274

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

Lab Report Status
Marks: …………………………………… **Signature: ....**
Comments: .... **Date: ....**

TITLE: Implementation of loop using assembly language.

1. INTRODUCTION

This lab helps us understand how loops work in assembly language programming. A loop allows a set of instructions to repeat several times until a specific condition is met. In 8086 assembly, the LOOP instruction automatically decreases the CX register and continues to repeat the program until CX becomes zero. By using loops, we can easily perform repetitive calculations such as finding the sum of numbers or calculating factorials. This lab gives hands-on practice with the use of loops, helping learners improve their understanding of control flow and program logic in microprocessor programming.

2. OBJECTIVES

- To learn how to use loop instructions in 8086 assembly language programs.
- To understand how the CX register controls the repetition of instructions in a loop.
- To develop problem-solving skills by implementing loop-based assembly programs for mathematical operations.

3. PROCEDURE

Problem-1: Take a number n from user. After that find out the factorial of that number n. (Suppose for n=5, you have to find out factorial= 1 × 2 × 3 × 4 × 5.

Problem-1 Procedure:

1. Initialize the data segment using MOV AX, @DATA and MOV DS, AX.
2. Display the message “Enter the number:” on the screen.
3. Take a single-digit number as input and store it in N.
4. Set RESULT = 1 and load N into the loop counter CX.
5. Multiply RESULT by CX in each loop until CX becomes zero.
6. After the loop ends, display the message “Factorial is:”.
7. Divide the result by 10 to get tens and one’s digits.
8. Convert digits to ASCII and print them on the screen.
9. Exit the program.

Problem-2: Implement a loop to find out the summation of 1^2 + 2^2 + 3^2 + ... + n^2. You can take n from user as an input.

Problem-2 Procedure:

1. Initialize the data segment using
   MOV AX, @DATA and MOV DS, AX.
2. Display the message “Enter the number:” on the screen.
3. Take a single-digit number from the user and store it in variable N.
4. Set SUM = 0 to store the total of squares.
5. Load N into the loop counter (CX) to control how many times the loop runs.
6. In each loop iteration:
   - Copy the current counter value into AL.
   - Multiply AL by itself (MUL AL) to get the square of the number.
   - Add this square to SUM.
7. Continue looping until CX becomes zero.
8. After the loop ends, display the message “Summation of squares is:”.
9. Divide SUM by 10 to separate the tens digit (quotient) and one’s digit (remainder).
10. Convert both digits to ASCII by adding 30H and print them one after another on the screen.
11. Exit the program.

4. IMPLEMENTATION

Problem-1: Source Code:

```
.MODEL SMALL
.STACK 100H
.DATA
N DB ?
STR DB "Enter the number: $"
MSG DB 0DH,0AH,"Factorial is: $"
RESULT DW 1

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

   ; Display prompt
   MOV DX, OFFSET STR
   MOV AH,9
INT 21H

; Read user input
MOV AH,1
INT 21H
SUB AL,30H
MOV N,AL

; Initialize RESULT = 1
MOV AX,1
MOV RESULT,AX

; Load counter CX = N
MOV CL,N
MOV CH,0
```

```
FACTORIAL_LOOP:
   MOV AX, RESULT
   MUL CX         ; AX = RESULT *
CX
   MOV RESULT, AX
   LOOP FACTORIAL_LOOP

   ; Print "Factorial is: "
   MOV DX, OFFSET MSG
   MOV AH,9
   INT 21H

   ; Print factorial (two digits max)
   MOV AX, RESULT
   MOV BL,10
   DIV BL         ; AX / 10, quotient
in AL, remainder in AH
   MOV BH,AL      ; tens digit
   MOV BL,AH      ; units digit
```

```
   ; Print tens digit
   MOV DL,BH
   ADD DL,30H
   MOV AH,2
   INT 21H

   ; Print units digit
   MOV DL,BL
   ADD DL,30H
   MOV AH,2
   INT 21H

   ; Exit
   MOV AH,4CH
   INT 21H

MAIN ENDP
END MAIN
```

Problem-2 Source Code:

```
.MODEL SMALL
.STACK 100H
.DATA
N DB ?
STR DB "Enter the number: $"
MSG DB 0DH,0AH,"Summation of squares is:
$"
SUM DW 0

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

   ; Display prompt
   MOV DX, OFFSET STR
   MOV AH, 9
   INT 21H

   ; Read user input (single digit)
   MOV AH, 1
```

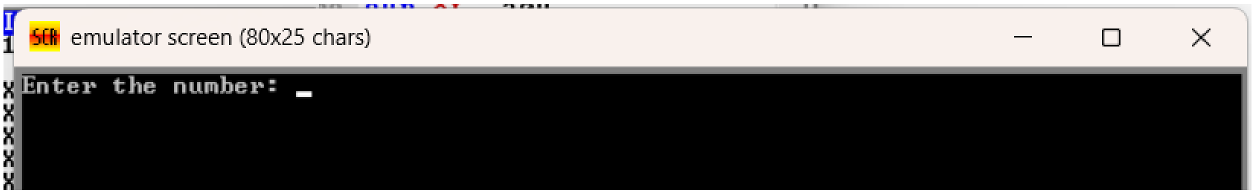
```
   INT 21H
   SUB AL, 30H
   MOV N, AL

   ; Initialize SUM = 0
   MOV AX, 0
   MOV SUM, AX

   ; Initialize counter CX = N
   MOV CL, N
   MOV CH, 0

SUM_LOOP:
   MOV AL, CL      ; current number
   MUL AL          ; AX = AL * AL = CL^2
   ADD SUM, AX     ; SUM += CL^2
   LOOP SUM_LOOP

   ; Display result message
   MOV DX, OFFSET MSG
   MOV AH, 9
```

```
    INT 21H

    ; Prepare to print result (two digits max)
    MOV AX, SUM
    MOV BL, 10
    DIV BL           ; AX / 10 ? AL = quotient
(tens), AH = remainder (units)
    MOV BH, AL       ; tens digit
    MOV BL, AH       ; ones digit

    ; Print tens digit
    MOV DL, BH
    ADD DL, 30H
    MOV AH, 2
    INT 21H

    ; Print ones digit
    MOV DL, BL
    ADD DL, 30H
    MOV AH, 2
    INT 21H

    ; Exit program
    MOV AH, 4CH
    INT 21H

MAIN ENDP
END MAIN
```

5. OUTPUT

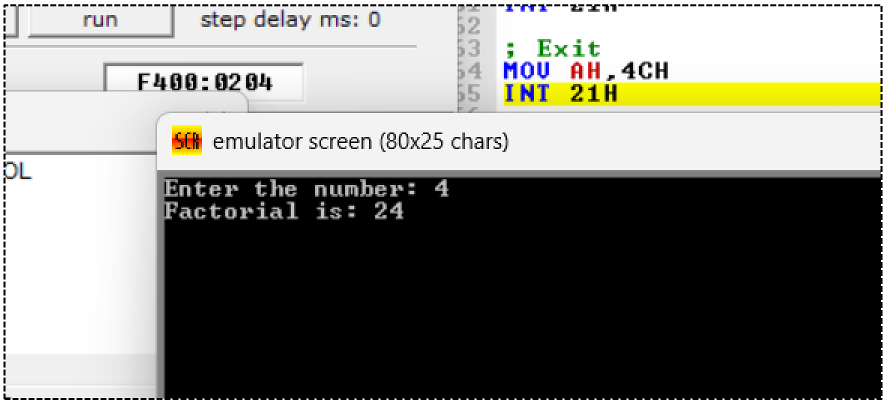
Output Problem-1

Step-1



```
Enter the number: _
```

Final Step

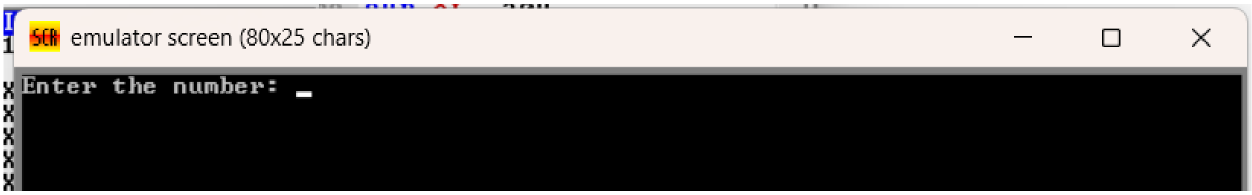


```
; Exit
MOV AH,4CH
INT 21H
```

```
Enter the number: 4
Factorial is: 24
```

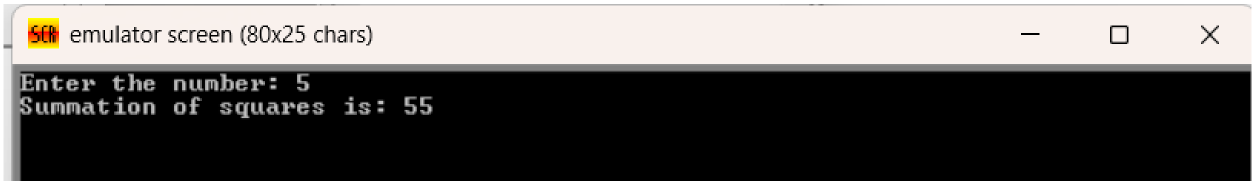
Output Problem-2

Step-1



```
Enter the number: _
```

Final Step



```
Enter the number: 5
Summation of squares is: 55
```

6. ANALYSIS AND DISCUSSION

In this lab, I learned how loops work in assembly language using the 8086 processor. The LOOP instruction helped repeat a block of code until a specific condition was met. The CX register was used as a counter that decreases automatically each time the loop runs. This made it easier to perform repeated operations such as adding numbers or calculating factorials.

By doing this experiment, I understood how assembly language handles repetition without using high-level statements. It also improved my understanding of how registers and instructions work together to control program flow and achieve accurate results.

.