

**LAB Report**

**COURSE TITLE –** Microprocessor Lab

**COURSE CODE –** CSE 360

***Submitted To***

***Md. Ismail***

*Lecturer of UITS*

***Submitted By***

*Md. Shoyaif Rahman (****0432310005101050****)*

***Semester:*** *Autumn-2025 (6th)*

***Department:*** *CSE*

***Batch:*** *53*

***Section:****6A2*

Date: 29/10/25

Assembly Language Lab Report

# Lab report: 07

# Experiment Name: Write an assembly code to check whether a string is palindrome or not.

# Process:

 **Initialize the Program:**

* Set up the data segment using MOV AX,@DATA and MOV DS,AX.

 **Define Data:**

* MSG contains the string to check, ending with $ as a string terminator.
* PAL stores the message "Palindrome$" to display if the string is a palindrome.
* NOTPAL stores the message "Not Palindrome$" to display if the string is not a palindrome.

 **Find the Length of the String:**

* Initialize CX=0 as a counter and SI=0 as the index to traverse the string.
* Loop through MSG:
  + Load the current character into AL using MOV AL, MSG[SI].
  + Compare it with $ (the string terminator).
  + If $ is found, jump to LEN\_FOUND.
  + Otherwise, increment CX and SI and repeat.
* After the loop, SI is decremented by 1 to point to the last character of the string.

 **Initialize Pointers for Comparison:**

* DI=0 points to the start of the string.
* SI points to the end of the string (last character).

 **Compare Characters from Start and End:**

* Start CHECK\_LOOP:
  + Load the character at the start (MSG[DI]) into AL.
  + Load the character at the end (MSG[SI]) into BL.
  + Compare AL and BL.
  + If they are not equal, jump to NOT\_PAL\_MSG (string is not a palindrome).
  + Increment DI (move start forward) and decrement SI (move end backward).
  + Continue looping until the start and end pointers meet or cross (DI <= SI).

 **Display Result if Palindrome:**

* If all characters matched, load DX with the offset of PAL.
* Use DOS interrupt 21H with AH=09H to display "Palindrome".

 **Display Result if Not Palindrome:**

* If a mismatch is found, load DX with the offset of NOTPAL.
* Use DOS interrupt 21H with AH=09H to display "Not Palindrome".

 **Exit Program:**

* Terminate the program using DOS interrupt 21H with AH=4CH.

## 2. Implementation (Program Code – ASM)

.MODEL SMALL

.STACK 100H

.DATA

MSG DB 'LEVEL$'

PAL DB 'Palindrome$',0DH,0AH

NOTPAL DB 'Not Palindrome$',0DH,0AH

.CODE

MAIN PROC

MOV AX,@DATA

MOV DS,AX

MOV CX,0

MOV SI,0

FIND\_LEN:

MOV AL, MSG[SI]

CMP AL,'$'

JE LEN\_FOUND

INC CX

INC SI

JMP FIND\_LEN

LEN\_FOUND:

DEC SI

MOV DI,0

CHECK\_LOOP:

MOV AL, MSG[DI]

MOV BL, MSG[SI]

CMP AL, BL

JNE NOT\_PAL\_MSG

INC DI

DEC SI

CMP DI, SI

JLE CHECK\_LOOP

MOV DX, OFFSET PAL

MOV AH,09H

INT 21H

JMP EXIT\_PROGRAM

NOT\_PAL\_MSG:

MOV DX, OFFSET NOTPAL

MOV AH,09H

INT 21H

EXIT\_PROGRAM:

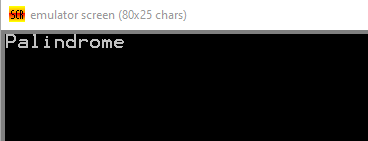
MOV AH,4CH

INT 21H

MAIN ENDP

END MAIN

**output:**

****

### 3. Result

The program checks whether the string 'LEVEL' is a palindrome by comparing characters from the start and end of the string using a loop. If all corresponding characters match, it prints "Palindrome"; otherwise, it prints "Not Palindrome". The program demonstrates the use of loops, string traversal from both ends, conditional jumps for decision-making, and DOS interrupt 21H for displaying messages on the screen.

### 4. Conclusion

The program effectively demonstrates how to check a string for palindrome properties in 8086 assembly language. It shows sequential memory access using index registers SI and DI, loop control for repeated character comparison, and conditional branching using CMP and JNE. This example reinforces fundamental assembly language concepts, including string traversal, comparison operations, loop management, and DOS-based character output.