Computer Organization & Assembly Language

Project Report

Submitted by:

Muhammad Abdullah	24-CS-172
Amin bin Rashid	24-CS-157
Rubbani Gul	24-CS-052

Submitted to:

Ma'am Shamshad Bibi

Department of Computer Science
HITEC University, Taxila

String Reversal and Palindrome Checker

Abstract:

This project implements a string reverse and palindrome checking system using 8086 Assembly Language. The program takes a string input from the user, displays the original string, reverses the string using pointer-based logic, and then checks whether the string is a palindrome. The project demonstrates low-level string manipulation, memory handling, and control flow using DOS interrupts and assembly language procedures.

Introduction:

Assembly language provides direct interaction with hardware and memory, making it essential for understanding low-level computing concepts. String manipulation in assembly language requires careful handling of memory addresses, registers, and loops.
This project focuses on implementing two common string operations string reversal and palindrome checking using the 8086 microprocessor architecture and DOS interrupts.

Problem Statement:

To design and implement an assembly language program that:

1. Accepts a string input from the user.
2. Displays the original string.
3. Reverse the string using pointer manipulation.
4. Check whether the string is a palindrome.
5. Displays appropriate output messages.

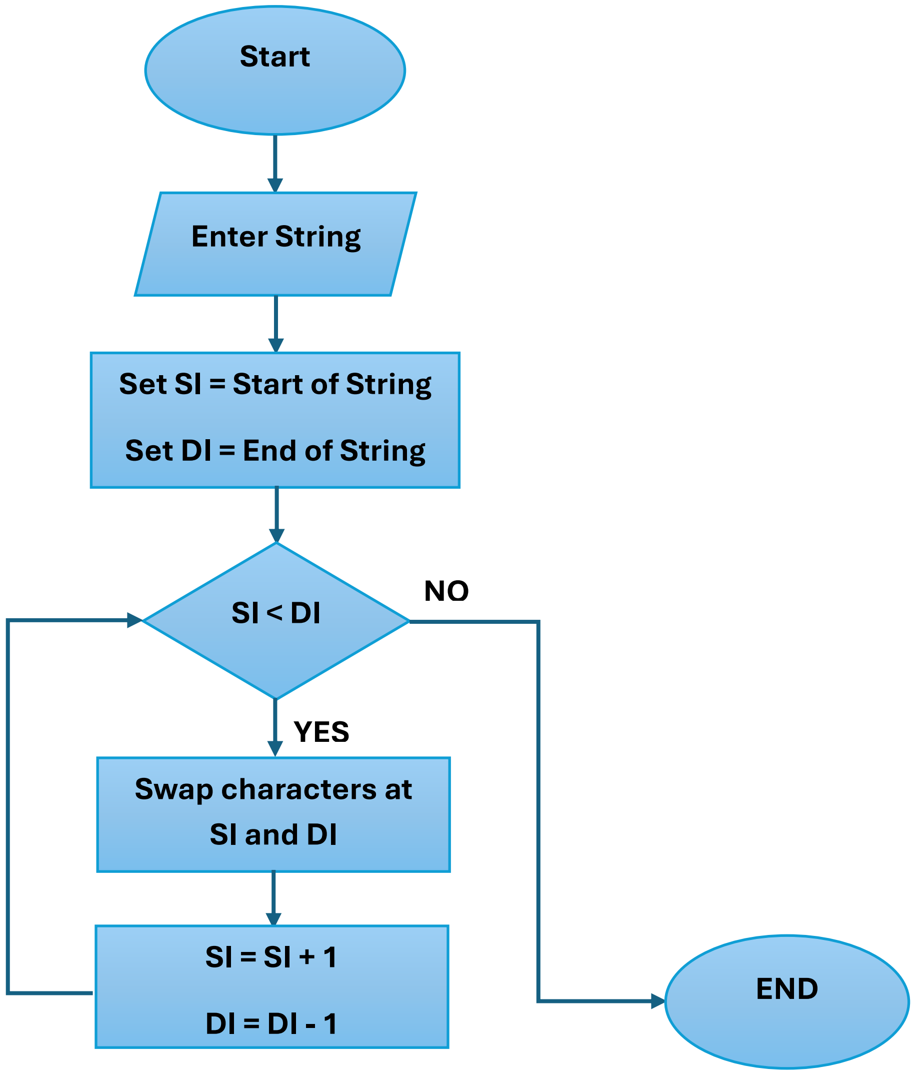
System Design / Flowchart

System Design Overview:

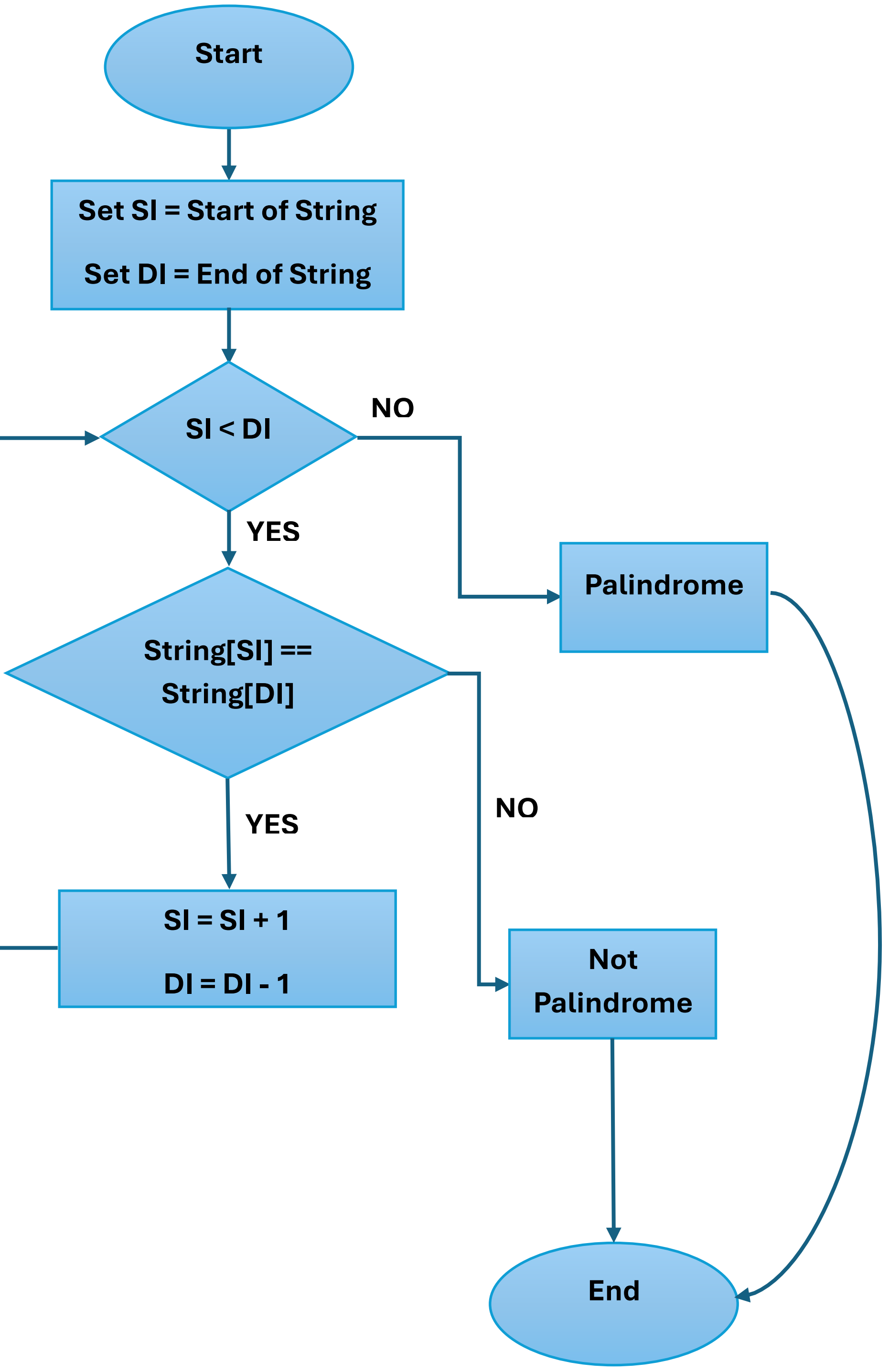
- Input is taken using DOS interrupt INT 21h (AH=0Ah)
- String is stored in a buffered format
- $ is used as a string terminator for printing
- Two separate procedures are used:
    - reverse
    - palindrome

Flowchart:

For Reverse:



For Palindrome:



Methodology:

1. DOS buffered input is used to read the string.
2. $ terminator is added after input to avoid garbage output.
3. Two pointers (**SI** and **DI**) are used:
   - SI points to the start of the string.
   - DI points to the end of the string.
4. The reverse procedure swaps characters until pointers meet.
5. The palindrome procedure compares characters from both ends.
6. Procedures are called using **CALL** and returned using **RET**.

Algorithms & Assembly Code Explanation

String Reverse Algorithm:

1. Initialize SI to start of string.
2. Initialize DI to last character of string.
3. While SI < DI:
   - Swap characters at SI and DI.
   - Increment SI
   - Decrement DI
4. Stop when pointers meet or cross.

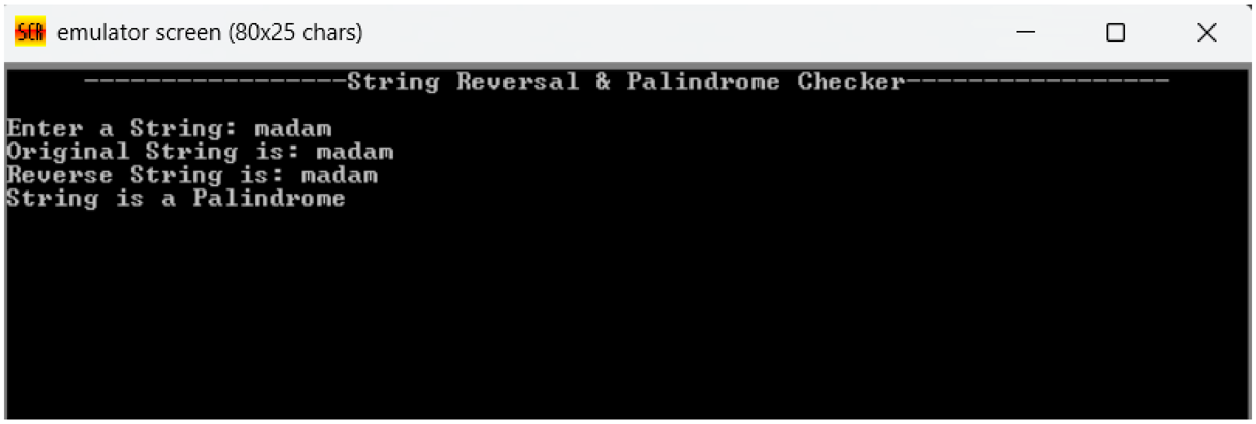
Palindrome Checking Algorithm:

1. Initialize SI and DI similarly.
2. While SI < DI:
   - Compare characters at SI and DI.
   - If mismatch → Not Palindrome.

   - Else move pointers inward.

3. If all characters match → Palindrome.

Screenshots of Working Program:

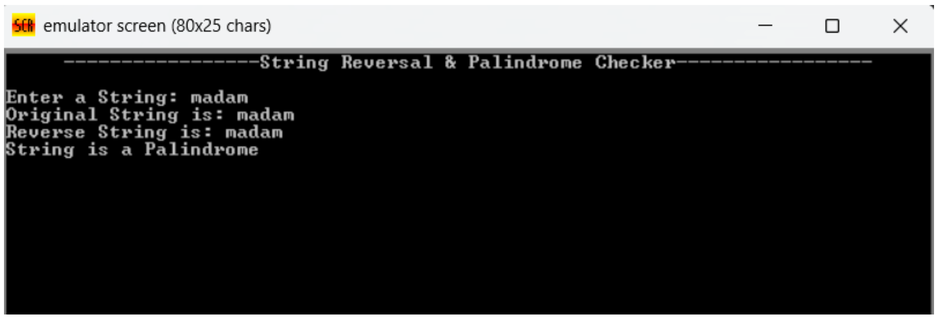


emulator screen (80x25 chars)

```
     -----------------String Reversal & Palindrome Checker-----------------

Enter a String: madam
Original String is: madam
Reverse String is: madam
String is a Palindrome
```

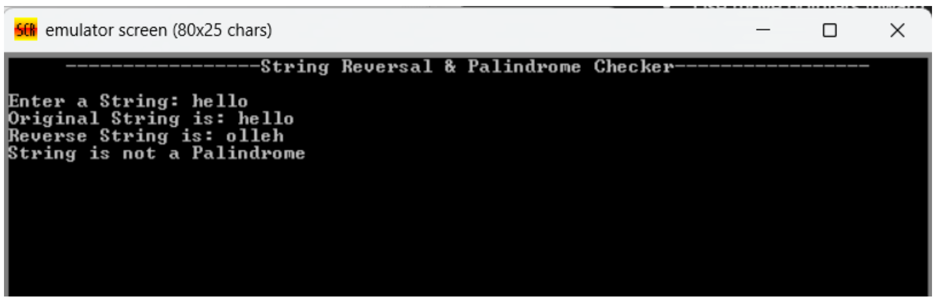
Results / Output:



emulator screen (80x25 chars)

```
     -----------------String Reversal & Palindrome Checker-----------------

Enter a String: madam
Original String is: madam
Reverse String is: madam
String is a Palindrome
```



emulator screen (80x25 chars)

```
     -----------------String Reversal & Palindrome Checker-----------------

Enter a String: hello
Original String is: hello
Reverse String is: olleh
String is not a Palindrome
```

Conclusion:

This project successfully demonstrates string manipulation using 8086 Assembly Language. By using pointer-based logic and modular procedures, the program efficiently reverses a string and checks for palindrome conditions. The project enhances understanding of low-level memory operations, register usage, and DOS interrupt handling, which are fundamental concepts in assembly language programming.