Name:	Rahul Yadav
Roll No:	76
Class/Sem:	SE/IV
Experiment No.:	7
Title:	Program to find whether given string is palindrome or not
Date of Performance:	24/2/24
Date of Submission:	6/3/24
Marks:	
Sign of Faculty:	

Aim: Assembly Language Program to find given string is Palindrome or not.

Theory:

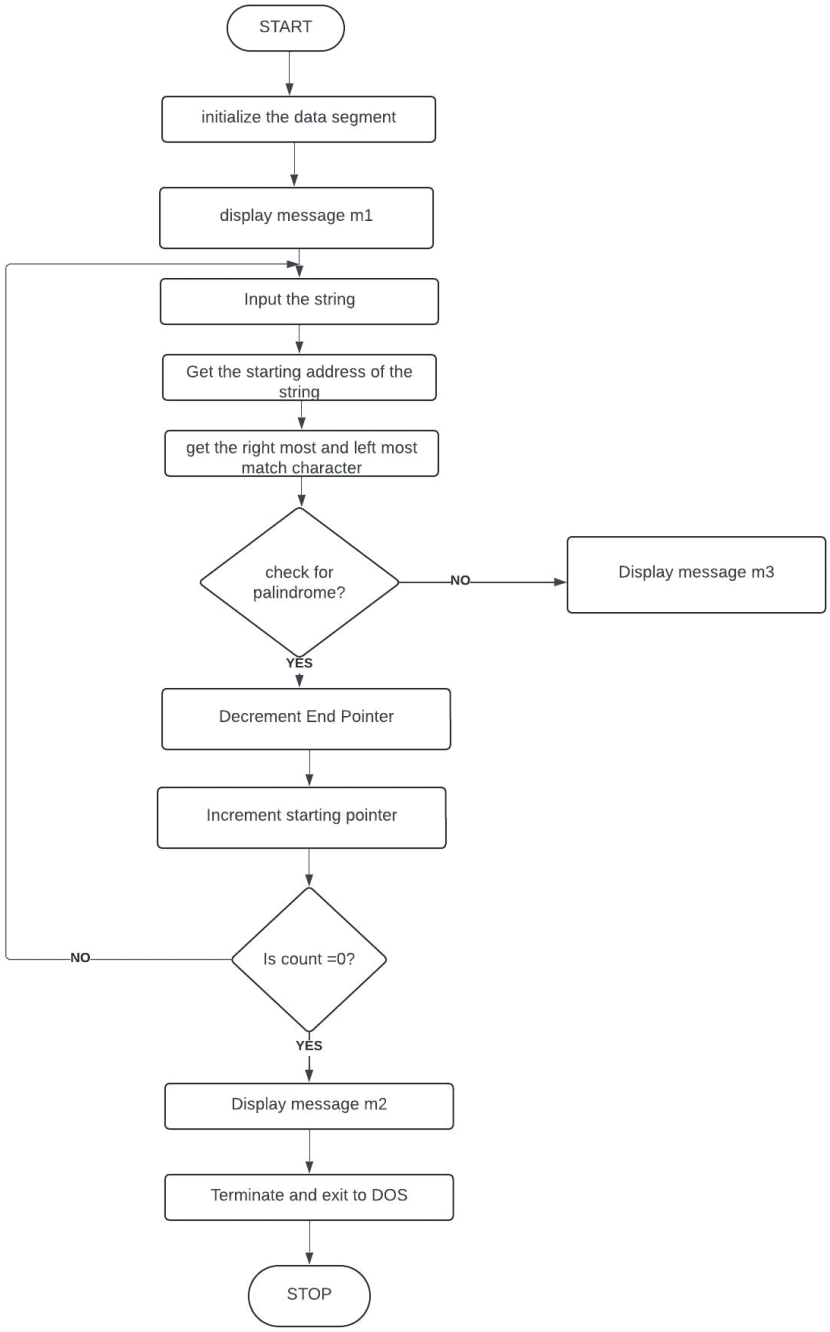
A palindrome string is a string when read in a forward or backward direction remains the same. One of the approach to check this is iterate through the string till middle of the string and compare the character from back and forth.

Algorithm:

- Initialize the data segment.
- Display the message M1
- Input the string
- Get the string address of the string
- Get the right most character
- Get the left most character

- Check for palindrome.
- If not Goto step 14
- Decrement the end pointer
- Increment the starting pointer.
- Decrement the counter
- If count not equal to zero go to step 5
- Display the message m2
- Display the message m3
- To terminate the program using DOS interrupt
    - Initialize AH with 4ch
    - Call interrupt INT 21h
- Stop

flowchart:

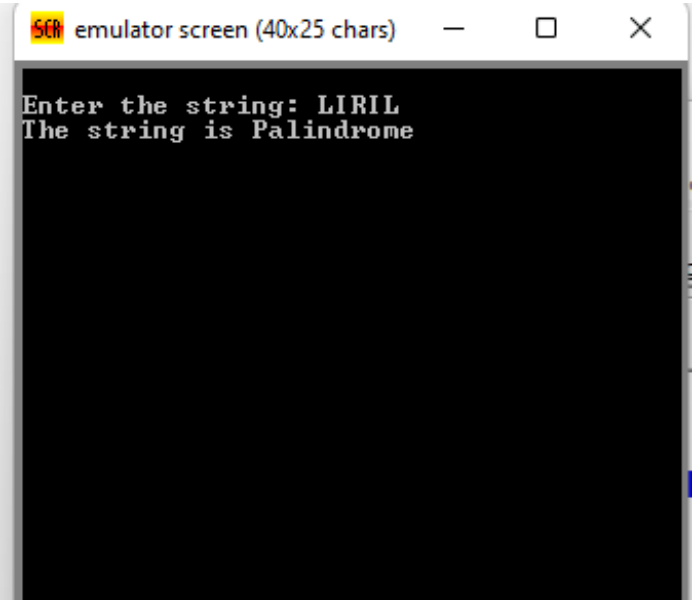


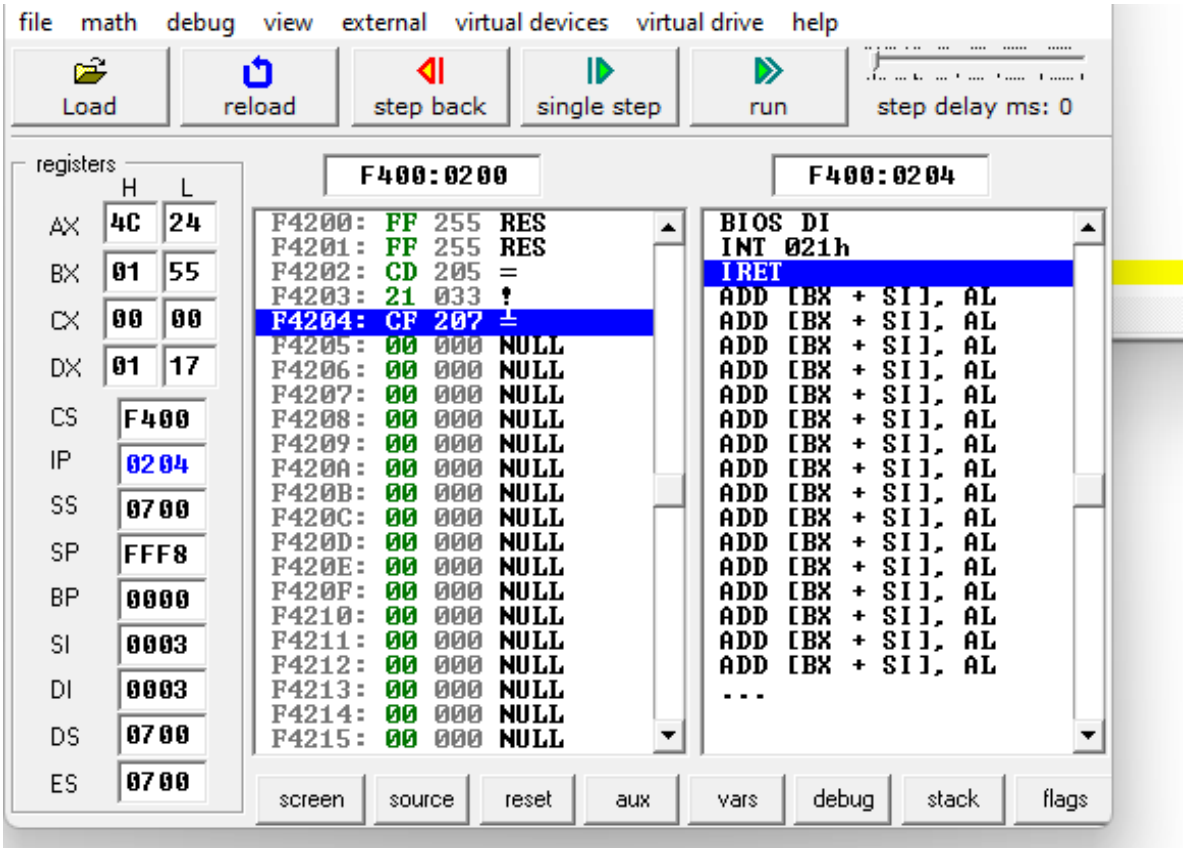
code& output:

```
org 100h

.data

m1 db 10,13,'Enter the string: $'
```

```
m2 db 10,13,'The string is Palindrome$'

m3 db 10,13,'The string is not a  Palindrome$'

buff db 80

.code

lea dx,m1

mov ah,09h

int 21h

lea dx,buff

mov ah,0ah

int 21h

lea bx,buff+1

mov si,01h

mov ch,00h

mov cl,[buff+1]

mov di,cx

Sar cl,1

pal:

mov ah,[buff+si]

mov al,[buff+di]

cmp al,ah

JC L1

inc si

dec di

loop pal

lea dx,m2
```

```
mov ah,09h

int 21h

JMP L2

L1:

lea dx,m3

mov ah,09h

int 21h

JMP L2

L2:

mov ah,4ch

int 21h
```





Conclusion:

In conclusion, the program to determine whether a given string is a palindrome or not is a fundamental exercise in programming that demonstrates basic string manipulation and logical reasoning. Palindromes are strings that read the same forwards and backwards, and the program typically involves comparing characters from the beginning and end of the string to check for equality.

- Explain SAR INSTRUCTION

The SAR (Shift Arithmetic Right) instruction is a fundamental operation in computer architecture used to perform a bitwise arithmetic right shift on binary data. It's commonly found in assembly language and low-level programming contexts. Here's an explanation of how it works:

Basic Functionality: SAR shifts the bits of a binary number to the right by a specified number of positions while preserving the sign bit. It treats the leftmost bit (most significant bit) as the sign bit, maintaining the sign of the original number after shifting.

Syntax: The SAR instruction typically takes two operands: the destination operand (the data to be shifted) and the count operand (the number of positions to shift by).

For each shift operation, SAR moves each bit of the operand to the right by the specified count, filling the vacant positions with the original value of the sign bit to preserve the sign of the number.

The original leftmost bit (sign bit) is duplicated into the vacated positions as the shift occurs.

If the sign bit is 0 (indicating a positive number), SAR fills the vacant positions with 0s. If the sign bit is 1 (indicating a negative number), SAR fills the vacant positions with 1s.

The value of the sign bit after shifting depends on the original value and whether the shift count is greater than the number of bits in the operand.

Example:

Let's say we have the binary number 10101010 (8 bits) and we perform a SAR operation with a count of 1.

Before the shift: 10101010

After the shift: 11010101

Explanation: Each bit is shifted one position to the right, and the sign bit (1 in this case) is duplicated into the vacated position.

Applications:

SAR is commonly used in low-level programming for tasks such as division by powers of 2, signed number manipulation, and implementing arithmetic operations in microprocessors.

It's essential for tasks where bitwise operations are necessary, particularly in systems programming, device driver development, and other low-level programming tasks.

- Explain DAA instruction.

The DAA (Decimal Adjust for Addition) instruction is a processor instruction found in many assembly languages, especially those used in Intel x86 architecture. Its primary purpose is to adjust the result of an addition operation performed on binary-coded decimal (BCD) numbers to ensure that the result is properly formatted in BCD.

Context: BCD is a binary encoding of decimal numbers where each decimal digit is represented by a 4-bit binary number. BCD numbers are often used in applications where decimal arithmetic is required, such as financial calculations.

Basic Functionality: After an addition operation involving BCD numbers, the result may not be properly formatted in BCD. The DAA instruction corrects this by adjusting the result to ensure it represents a valid BCD number

DAA operates on the accumulator register (typically AL or AX in x86 architecture) after an addition operation, adjusting the contents of the accumulator to ensure it represents a valid BCD result.

DAA examines the least significant 4 bits (nibble) and the most significant 4 bits of the accumulator separately.

It checks each nibble for values greater than 9 or if it has the auxiliary carry flag set (indicating a carry from the lower nibble to the higher nibble).

If a nibble value is greater than 9 or if the auxiliary carry flag is set, DAA adds 6 to that nibble to correct it for BCD representation.

If the higher nibble needs adjustment due to the carry from the lower nibble, DAA adds 6 to the higher nibble.

After adjustment, if the resulting value is greater than 15, DAA sets the carry flag to indicate a carry out of the most significant nibble.

Example:

Let's say we have the BCD number 0x59 in the AL register (01011001 in binary). After adding another BCD number to it, the result may exceed 0x99.

Before DAA: AL = 0x59 (01011001)

After DAA: AL = 0x65 (01100101)

Explanation: DAA adjusts the least significant nibble (9) to 5, and the most significant nibble (5) to 6, ensuring that the result is a valid BCD number.

Applications:

DAA is commonly used in applications where BCD arithmetic is required, such as financial software, calculators, and embedded systems.

It's a critical instruction for performing accurate arithmetic operations on BCD numbers in assembly language programming.