Name:	Pranav Shetty
Roll No:	53
Class/Sem:	SE/IV
Experiment No.:	6
Title:	To perform program to reverse the word in string
Date of Performance:	27/02/2024
Date of Submission:	05/03/2024
Marks:	
Sign of Faculty:	

Aim: Assembly Language Program to reverse the word in string.

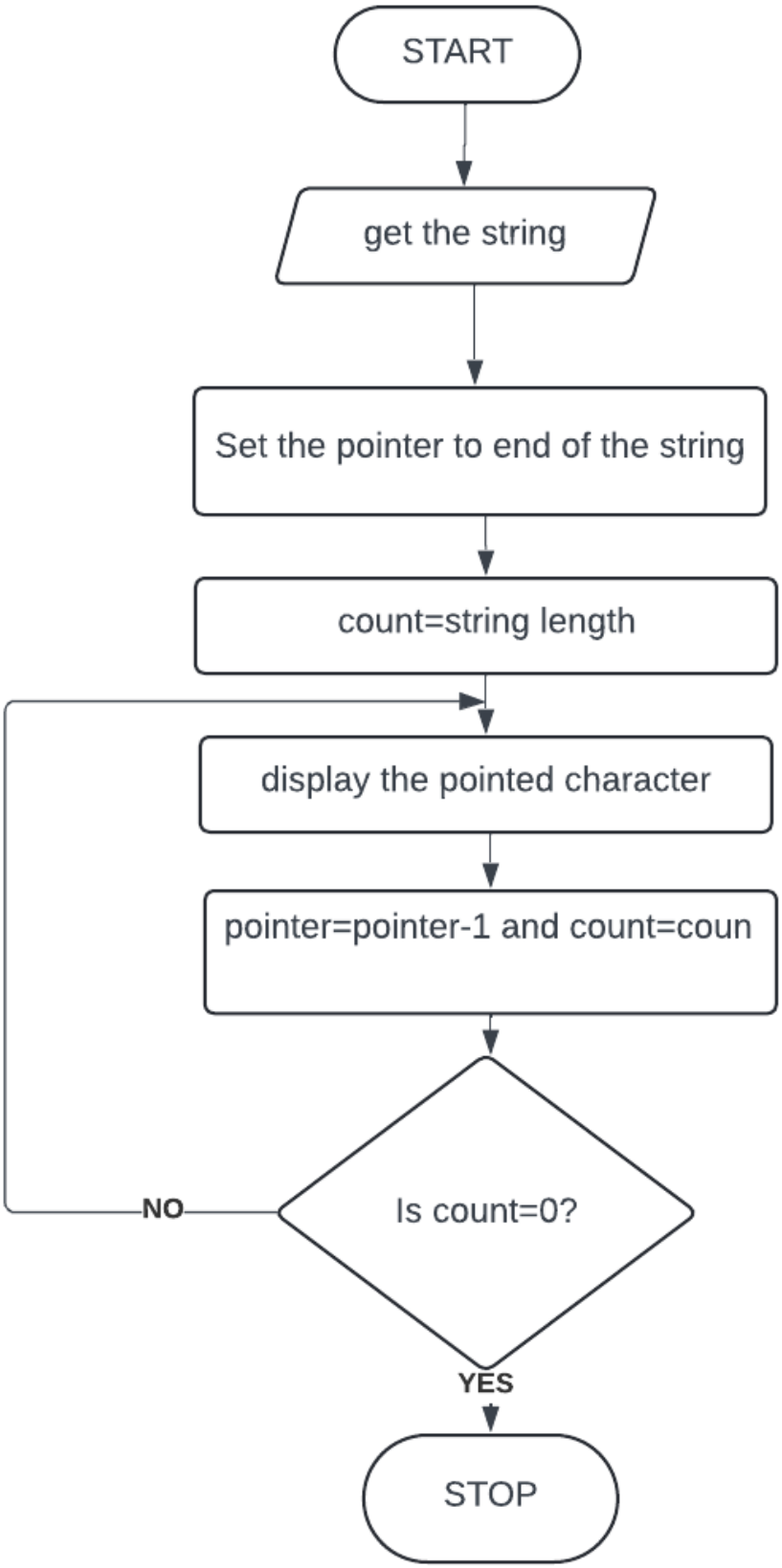
Theory:

This program will read the string entered by the user and then reverse it. Reverse a string is the technique that reverses or changes the order of a given string so that the last character of the string becomes the first character of the string and so on.

Algorithm:

1. Start
2. Initialize the data segment
3. Display the message -1
4. Input the string
5. Display the message 2
6. Take characters count in DI
7. Point to the end character and read it
8. Display the character
9. Decrement the count
10. Repeat until the count is zero
11. To terminate the program using DOS interrupt
    a. Initialize AH with 4ch
    b. Call interrupt INT 21h
12. Stop

Flowchart:


START
get the string
Set the pointer to end of the string
count=string length
display the pointed character
pointer=pointer-1 and count=coun
Is count=0?
NO
YES
STOP

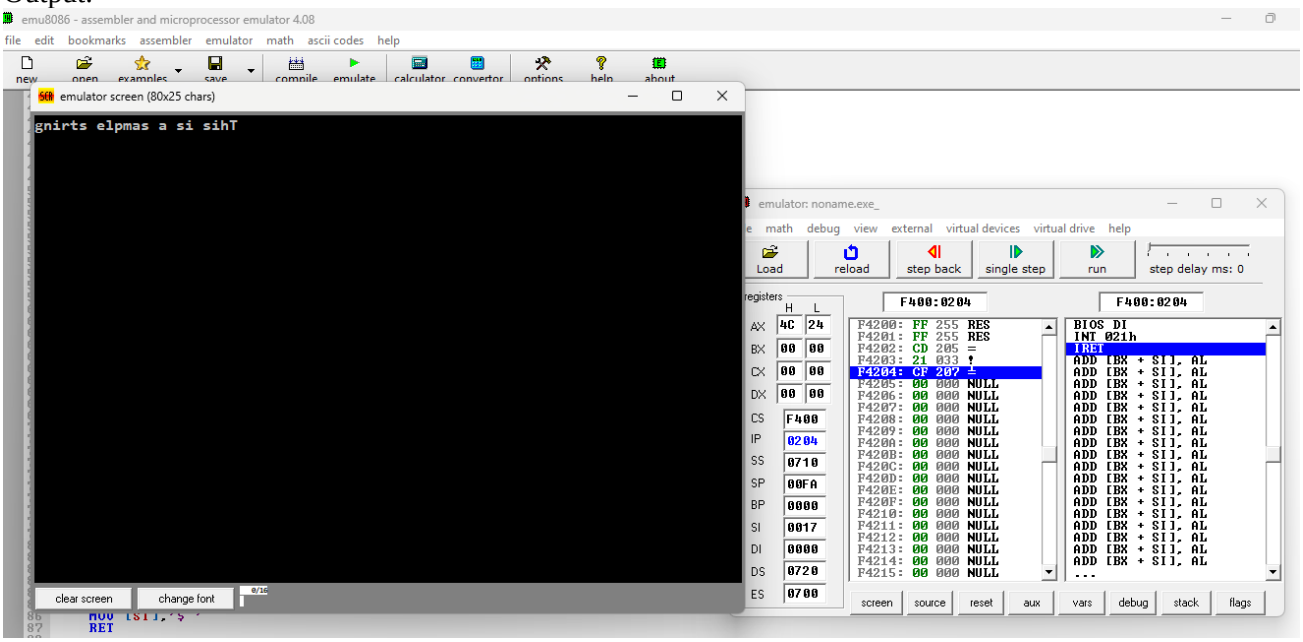
Code:

```
.MODEL SMALL
.STACK 100H
.DATA

; The string to be printed
STRING DB 'This is a sample string', '$'

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

; call reverse function
CALL REVERSE

; load address of the string
LEA DX,STRING

; output the string
; loaded in dx
MOV AH, 09H
INT 21H

; interrupt to exit
MOV AH, 4CH
INT 21H

MAIN ENDP
REVERSE PROC
    ; load the offset of
    ; the string
    MOV SI, OFFSET STRING

    ; count of characters of the;
    ;string
    MOV CX, 0H

    LOOP1:
    ; compare if this is;
    ;the last character
    MOV AX, [SI]
    CMP AL, '$'
    JE LABEL1
```

```
    ; else push it in the;
    ;stack
    PUSH [SI]

    ; increment the pointer;
    ;and count
    INC SI
    INC CX

    JMP LOOP1

    LABEL1:
    ; again load the starting;
    ;address of the string
    MOV SI, OFFSET STRING

        LOOP2:
        ;if count not equal to zero
        CMP CX,0
        JE EXIT

        ; pop the top of stack
        POP DX

        ; make dh, 0
        XOR DH, DH

        ; put the character of the;
        ;reversed string
        MOV [SI], DX

        ; increment si and;
        ;decrement count
        INC SI
        DEC CX

        JMP LOOP2


    EXIT:
    ; add $ to the end of string
    MOV [SI],'$ '
    RET

REVERSE ENDP
```

```
END MAIN
```

Output:



Conclusion:

1. Explain the difference between XLAT and XLATB

XLAT and XLATB are both x86 assembly language instructions used for byte translation. However, they have some differences:

1. XLAT (or XLATE):

- XLAT stands for "Translate Byte", and it is used to translate a byte in the AL register using a lookup table.
- The lookup table is pointed to by the DS:BX register pair.
- After translation, the result is stored back in the AL register.
- This instruction is primarily used for simple byte translation tasks.

2. XLATB:

- XLATB stands for "Translate Byte at DS:EBX", and it is similar to XLAT but with a different operand.
- In XLATB, the lookup table is pointed to by the DS:EBX register pair, unlike XLAT where it's pointed by DS:BX.
- The result of the translation is again stored back in the AL register.

2. Explain the instruction LAHF.

The `LAHF` instruction, which stands for Load AH from Flags, is an assembly language instruction in x86 architecture. Here's what it does:

1. Function: The `LAHF` instruction loads the contents of the flags register (EFLAGS) into the AH register. The AH register is the high byte of the AX register pair.

2. Flags Included: The flags loaded into AH are the six status flags and the two control flags. These are:

- SF (Sign Flag)
- ZF (Zero Flag)
- AF (Adjust Flag)
- PF (Parity Flag)
- CF (Carry Flag)
- OF (Overflow Flag)
- DF (Direction Flag)
- IF (Interrupt Enable Flag)

3. Format: The format of the `LAHF` instruction is simply `LAHF`, with no operands.

4. Purpose: The primary purpose of `LAHF` is to facilitate the manipulation of flags within the program. By loading the flags into a register (AH), programmers can perform various operations on them, such as testing for specific conditions or modifying them directly.

5.Usage `LAHF` is often used in conjunction with the `SAHF` (Store AH into Flags) instruction. Programmers may use `LAHF` to save the current state of the flags, perform some operations, and then restore the flags to their original state using `SAHF`.

Here's a simple example illustrating the usage of `LAHF`:

```assembly
; Example code snippet
    LAHF        ; Load flags into AH
    PUSHF       ; Push flags onto stack
    ...
    POPF        ; Restore flags from stack
```

In this example, the `LAHF` instruction loads the current state of the flags into the AH register. Then, the `PUSHF` instruction pushes the contents of the flags register onto the stack for later retrieval. After some operations, the `POPF` instruction restores the flags from the stack. This sequence allows the program to manipulate the flags while preserving their original state.