FR. Conceicao Rodrigues College of Engineering
Department of Computer Engineering

9. TO CHECK IF THE ENTERED STRING IS PALINDROME OR NOT

1. Course, Subject & Experiment Details

Academic Year	**2023-24**	**Estimated Time**	**Experiment No. 9– 02 Hours**
Course & Semester	**S.E. (Comps) – Sem. IV**	**Subject Name**	**Microprocessor**
Chapter No.	**2**	**Chapter Title**	**Instruction Set and Programming**
Experiment Type	**Software**	**Subject Code**	**CSC405**

Rubrics

Timeline (2)	Practical Skill & Applied Knowledge (2)	Output (3)	Postlab (3)	Total (10)	Sign

2. Aim & Objective of Experiment

CHECK IF THE ENTERED STRING IS PALINDROME OR NOT

Objective : Read a string from user and check whether it is palindrome or not and display appropriate message.

3. Software Required

TASM Assembler

4 . Brief Theoretical Description

Pre-Requisites: 1. Knowledge of TASM directives.
2. Knowledge of DOS interrupts.
3. Knowledge of string instruction and MACRO

String instruction/ data transfer

a) LEA register, Source - Loads effective address (offset Address) of source in given register.
e.g. LEA BX, Total;

b) STOSB/STOSW (Store string) – It is used to store AL (or AX) into a memory location pointed by ES:DI. DI is incremented or decremented after transfer depending upon DF.

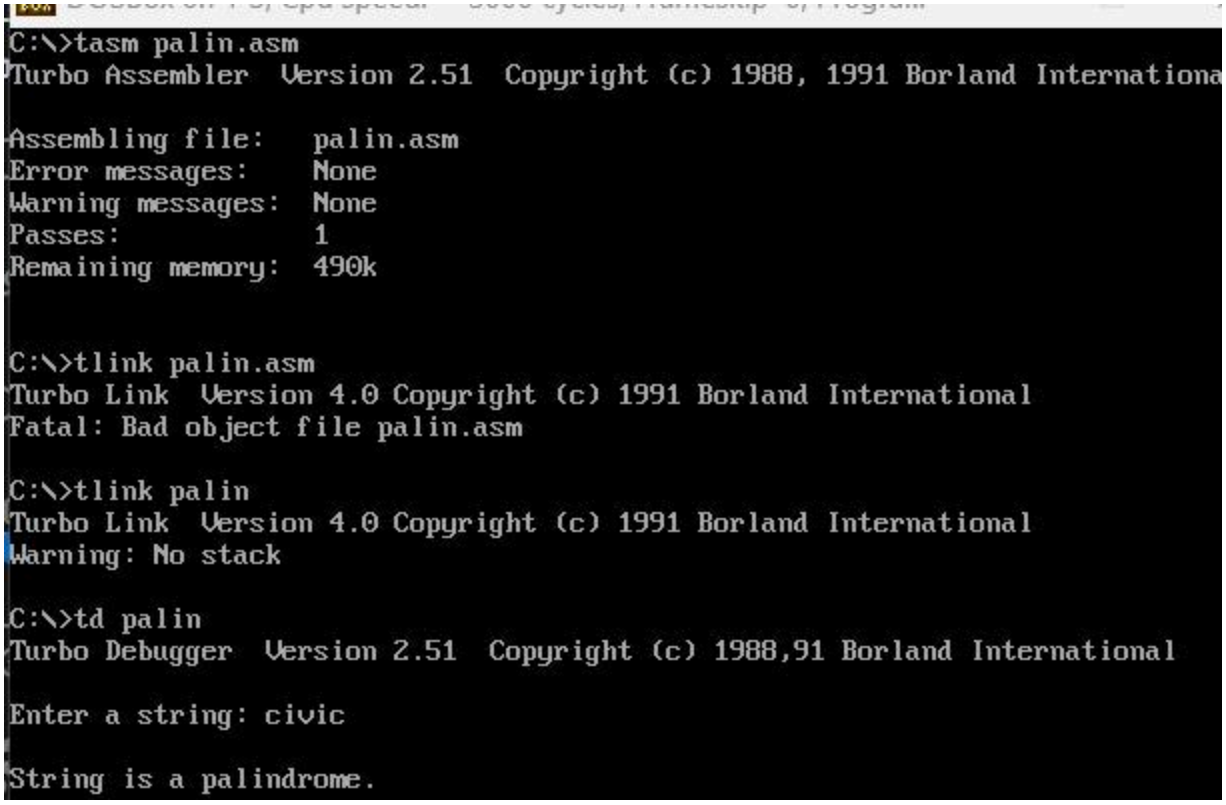
c) LODSB/LODSW: Used to copy the contents of memory location pointed by DS: SI and store it in AL registers. SI is incremented /decremented after transfer depending upon DF.

5. Algorithm:

1. Start.
2. Initialize Data segment (DS).
3. Initialize Extra segment (ES).
4. Ask user to enter a string.
5. Read each character of a string using function value 01h or 08h.
6. Store individual character read in a string S.
7. Make SI register point to first element of a string and DI resister to point to last element of a string.
8. Initialize count register to number of comparisons required.
9. Move contents pointed by SI to AL.
10. Compare character in AL with character pointed by DI.
11. If there is a mismatch (ZF=0) display a message “Not a palindrome” and stop.
12. If two characters are matching then increment SI, decrement DI, decrement count register.
13. If count register becomes zero display a message “String is palindrome” and stop.

palin.asm

```
.8086
.model small

print macro msg
    push ax
    mov ah, 09h
    lea dx, msg
    int 21h
    pop ax
endm

.data
    get db 10,13, "Enter a string: $"
    yes db 10,13, "String is a palindrome.$"
    no db 10,13, "String is not a palindrome.$"

    inp db 20 dup(' ')
    rev db 20 dup(' ')

.code
start:
    mov ax, @data
    mov ds, ax
    mov es, ax

    lea si, inp
    lea di, rev
    mov cx, 0000h

    print get

back:
    mov ah, 01h
    int 21h

    cmp al, 13
    jz next

    mov [si], al
    inc si
    inc cx
    jmp back

next:
    mov bl, cl

back1:
    dec si
    mov al, [si]
    mov [di], al
    inc di
    loop back1

    mov cl, bl

    lea si, inp
    lea di, rev

    repe cmpsb

    jz next1

    print no
    jmp exit

next1:
    print yes

exit:
    mov ah, 4ch
    int 21h

end start

```



Postlab:

Q1. Give the difference between Macro and procedure.

Macros:

- Execution: Occurs at assembly time, providing a form of text replacement.
- Parameterization: Highly flexible, allows passing parameters for reuse.
- Scope: Local to the source file where they are defined.
- Usage: Primarily for code expansion, especially for repetitive or boilerplate code.

Procedures:

- Execution: Occurs at runtime with a function call and return.
- Parameterization: Also supports parameter passing, but in a more structured manner.
- Scope: Can be global, allowing use across multiple files.
- Usage: Primarily for organizing code into reusable and modular functions or subroutines.

Q2.How many bytes are pushed onto the stack by a far CALL instruction? What do they represent?

A far CALL instruction in x86 assembly language pushes either 2 or 4 bytes onto the stack, depending on whether it uses a near or far addressing mode. These bytes represent the return address for control to resume after the called subroutine or function completes.

Near Addressing Mode:

- If a near address is used in the far 'CALL' instruction, the return address pushed onto the stack is the offset of the instruction following the 'CALL'.
- In this case, 2 bytes are pushed onto the stack.

Far Addressing Mode:

- If a far address is used in the far 'CALL' instruction, the return address pushed onto the stack consists of two parts:
    - The offset of the instruction following the 'CAll' (2 bytes).
    - The segment of the instruction following the 'CALL' (2 bytes).
- In this case, 4 bytes are pushed onto the stack.