NAME : Soham Manoj Pawar **Class : SE-IT**

ROLL NO : 71 **MPL LAB**

EXPERIMENT 10

AIM:
-

Check whether a given string is a palindrome or not.

LO No :- LO4

LO : Display assembly level programming using 8086 loop instruction set.

SOFTWARE :- Tasm Software

Theory :-

Assume CS : code , DS : data

- ASSUME statement can assign up to 4 segment registers in any sequences.
- DS: Data means that the assembler is to associate the name of data segment with DS register.

Similarly CS: Code tells the assembler to associate the name of code segment with CS register and so on.

MOV Ax,data MOV Ds,Ax :-

Initialize Ds to point start of memory, set alongside to store data.

LEA :-

Used to load the address of operand into the provided register. The LEA instruction is used to load a pointer into a register. It is actually an arithmetic instruction, and does not read RAM at all.

SHR :-

The SHR instruction is an abbreviation for ‘Shift Right’. This instruction simply shifts the mentioned bits in the register to the right side one by one by inserting the same number (bits that are being shifted) of zeroes from the left end. The rightmost bit that is being shifted is stored in the Carry Flag (CF).

```
Syntax:    SHR Register, Bits to be shifted
Example:   SHR AX, 2
```

NAME : Soham Manoj Pawar **Class : SE-IT**

ROLL NO : 71 **MPL LAB**

PRINT

Used to print the data or string present in object.

JMP :-

In the x86 assembly language, the JMP instruction performs an unconditional jump. Such an instruction transfers the flow of execution by changing the program counter.
i.e to jump to the provided address to proceed to the next instruction.

JZ :-

jz is "jump if zero". cmp subtracts its two operands, and sets flags accordingly.
(See here for reference.)
If the two operands are equal, the subtraction will result in zero and the ZF flag will be set.

JNZ :-

The jnz (or jne) instruction is a conditional jump that follows a test.
It jumps to the specified location if the Zero Flag (ZF) is cleared (0). jnz is commonly used to explicitly test for something not being equal to zero whereas jne is commonly found after a cmp instruction.

INC :-

Adds 1 to the destination operand, while preserving the state of the CF flag. The destination operand can be a register or a memory location. This instruction allows a loop counter to be updated without disturbing the CF flag.

DEC :-

Used to decrement the provided byte/word by 1. NPG − Used to negate each bit of the provided byte/word and add 1/2's complement. CMP − Used to compare 2 provided byte/word. AAS − Used to adjust ASCII codes after subtraction.

INT 21H :-

int 21h means, call the interrupt handler 0x21 which is the DOS Function dispatcher. the "mov ah,01h" is setting AH with 0x01, which is the Keyboard Input with Echo handler in the interrupt.
Syntax: int 21H
Example: int 21H

NAME : Soham Manoj Pawar **Class : SE-IT**

ROLL NO : 71 **MPL LAB**

Code :-

```
Assume CS: Code, DS: Data Data Segment m1
db0AH, 0DH,'ENTER THE STRING: $' m2 db
0AH,
0DH,'STRING IS PALINDROME$' m3 db 0AH,
0DH,'STRING IS NOT A PALINDROME$' Buff db
80H     db 00H     db 80H dup(0) Data
EndsPRINT MACRO MSG
MOV AH,
09HLEA DX,
MSG INT
21H ENDM
Code  Segment
Start: MOV AX,
Data
      MOV DS,
      AX PRINT
      M1 MOV
      AH, 0AH
      LEA DX,
      BUFF INT
      21H
      LEA BX,
      BUFF+2 MOV
      CH, 00H MOV
      CL, BUFF+1
      MOV DI, CX
      DEC DI
      MOV SI,
00HSHR CL,
01H
      Back: MOV AL,
      [BX+SI]MOV AH,
      [BX+DI] CMP AL,
      AH
      JNZ
      Last
      DEC DI
      INC SI
      DEC
      CX
      JNZ Back
      PRINT
      M2
      JMP Final
```

NAME : Soham Manoj Pawar **Class : SE-IT**

ROLL NO : 71 **MPL LAB**

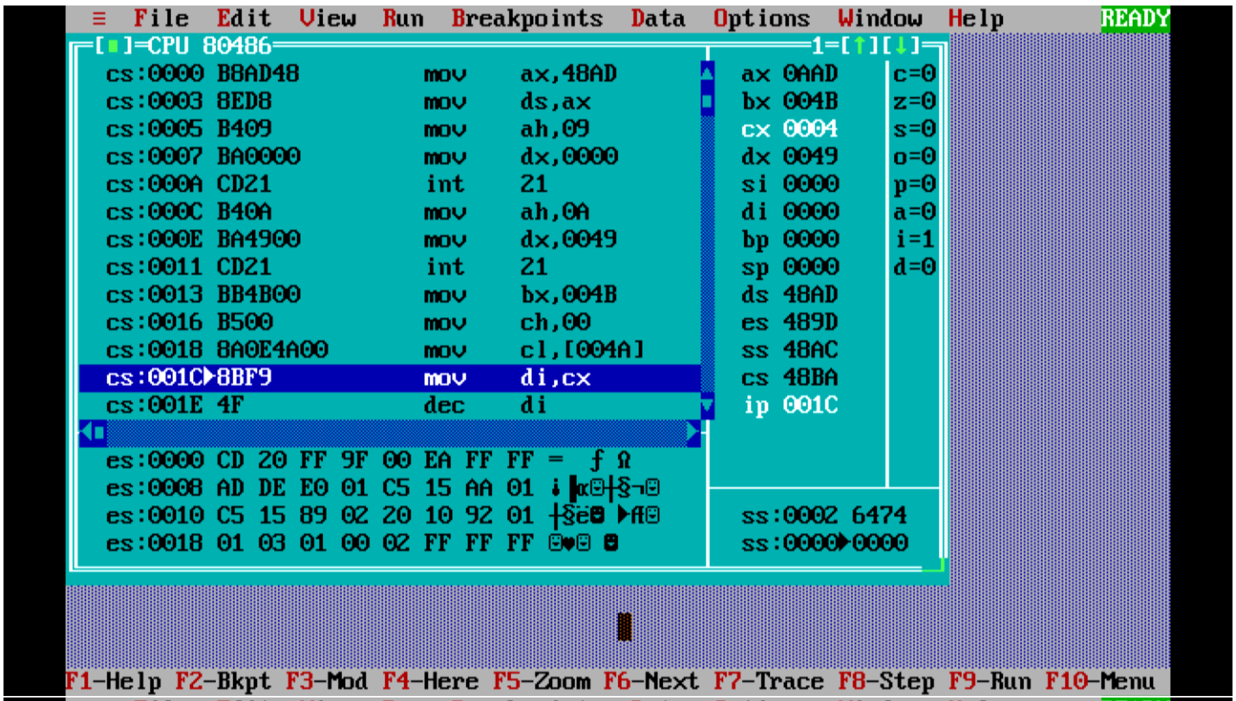
```
Last: PRINT M3
Final: MOV AH,
4CHINT 21H
```

NAME : Soham Manoj Pawar **Class : SE-IT**

ROLL NO : 71 **MPL LAB**

Code Ends
End Start

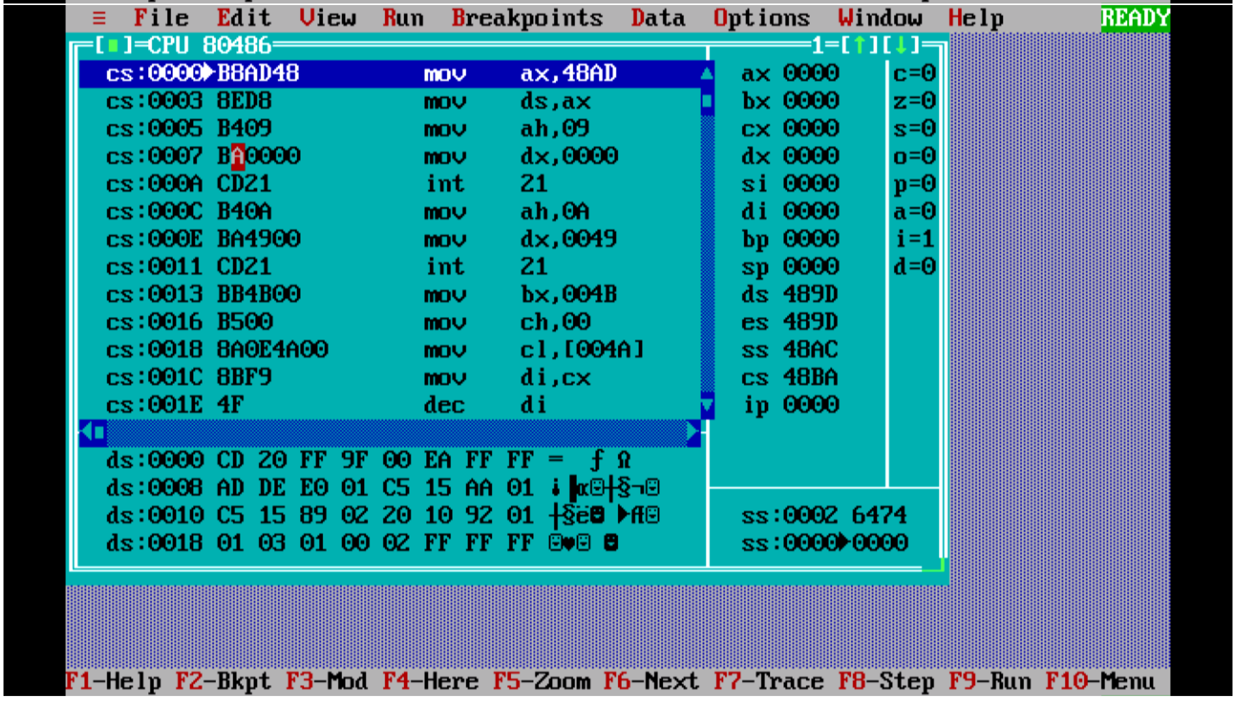
<u>Output :-</u>



```
 ≡  File  Edit  View  Run  Breakpoints  Data  Options  Window  Help        READY
[■]=CPU 80486===================================1=[↑][↓]=
  cs:0000 B8AD48         mov    ax,48AD          ax 0AAD   c=0
  cs:0003 8ED8           mov    ds,ax            bx 004B   z=0
  cs:0005 B409           mov    ah,09            cx 0004   s=0
  cs:0007 BA0000         mov    dx,0000          dx 0049   o=0
  cs:000A CD21           int    21               si 0000   p=0
  cs:000C B40A           mov    ah,0A            di 0000   a=0
  cs:000E BA4900         mov    dx,0049          bp 0000   i=1
  cs:0011 CD21           int    21               sp 0000   d=0
  cs:0013 BB4B00         mov    bx,004B          ds 48AD
  cs:0016 B500           mov    ch,00            es 489D
  cs:0018 8A0E4A00       mov    cl,[004A]        ss 48AC
  cs:001C►8BF9           mov    di,cx            cs 48BA
  cs:001E 4F             dec    di               ip 001C

  es:0000 CD 20 FF 9F 00 EA FF FF = ƒ Ω
  es:0008 AD DE E0 01 C5 15 AA 01 ¡ α☺┼§¬☺
  es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺        ss:0002 6474
  es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻           ss:0000►0000

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```



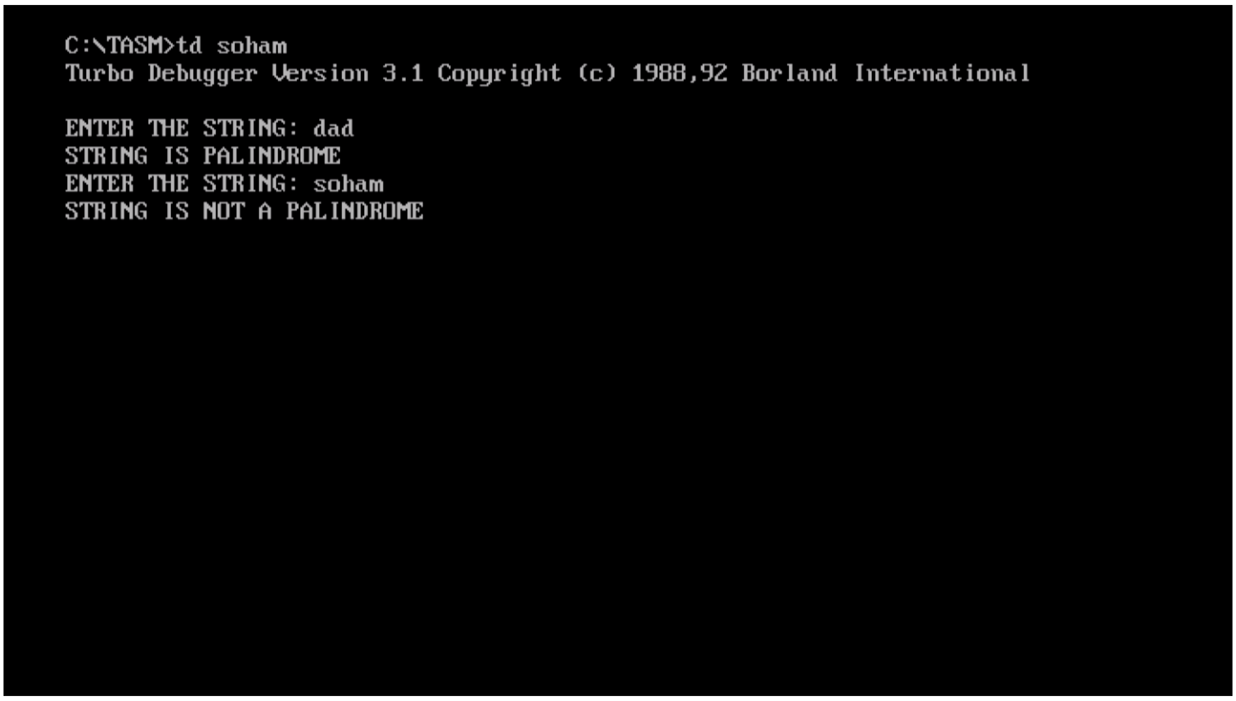
```
 ≡  File  Edit  View  Run  Breakpoints  Data  Options  Window  Help        READY
[■]=CPU 80486===================================1=[↑][↓]=
  cs:0000►B8AD48         mov    ax,48AD          ax 0000   c=0
  cs:0003 8ED8           mov    ds,ax            bx 0000   z=0
  cs:0005 B409           mov    ah,09            cx 0000   s=0
  cs:0007 BA0000         mov    dx,0000          dx 0000   o=0
  cs:000A CD21           int    21               si 0000   p=0
  cs:000C B40A           mov    ah,0A            di 0000   a=0
  cs:000E BA4900         mov    dx,0049          bp 0000   i=1
  cs:0011 CD21           int    21               sp 0000   d=0
  cs:0013 BB4B00         mov    bx,004B          ds 489D
  cs:0016 B500           mov    ch,00            es 489D
  cs:0018 8A0E4A00       mov    cl,[004A]        ss 48AC
  cs:001C 8BF9           mov    di,cx            cs 48BA
  cs:001E 4F             dec    di               ip 0000

  ds:0000 CD 20 FF 9F 00 EA FF FF = ƒ Ω
  ds:0008 AD DE E0 01 C5 15 AA 01 ¡ α☺┼§¬☺
  ds:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺        ss:0002 6474
  ds:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻           ss:0000►0000

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

NAME : Soham Manoj Pawar **Class : SE-IT**

ROLL NO : 71 **MPL LAB**



```
C:\TASM>td soham
Turbo Debugger Version 3.1 Copyright (c) 1988,92 Borland International

ENTER THE STRING: dad
STRING IS PALINDROME
ENTER THE STRING: soham
STRING IS NOT A PALINDROME
```

<u>Conclusion :</u>

Thus, we learnt how to check whether a string is palindrome or not using assembly language.