



Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science (AI&DS)

Name:	Yash Ravindra Kerkar
Roll No:	67
Class/Sem:	SE/IV
Experiment No.:	7
Title:	Program to find whether given string is palindrome or not
Date of Performance:	24/02/24
Date of Submission:	06/03/24
Marks:	
Sign of Faculty:	



Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science (AI&DS)

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

Theory:

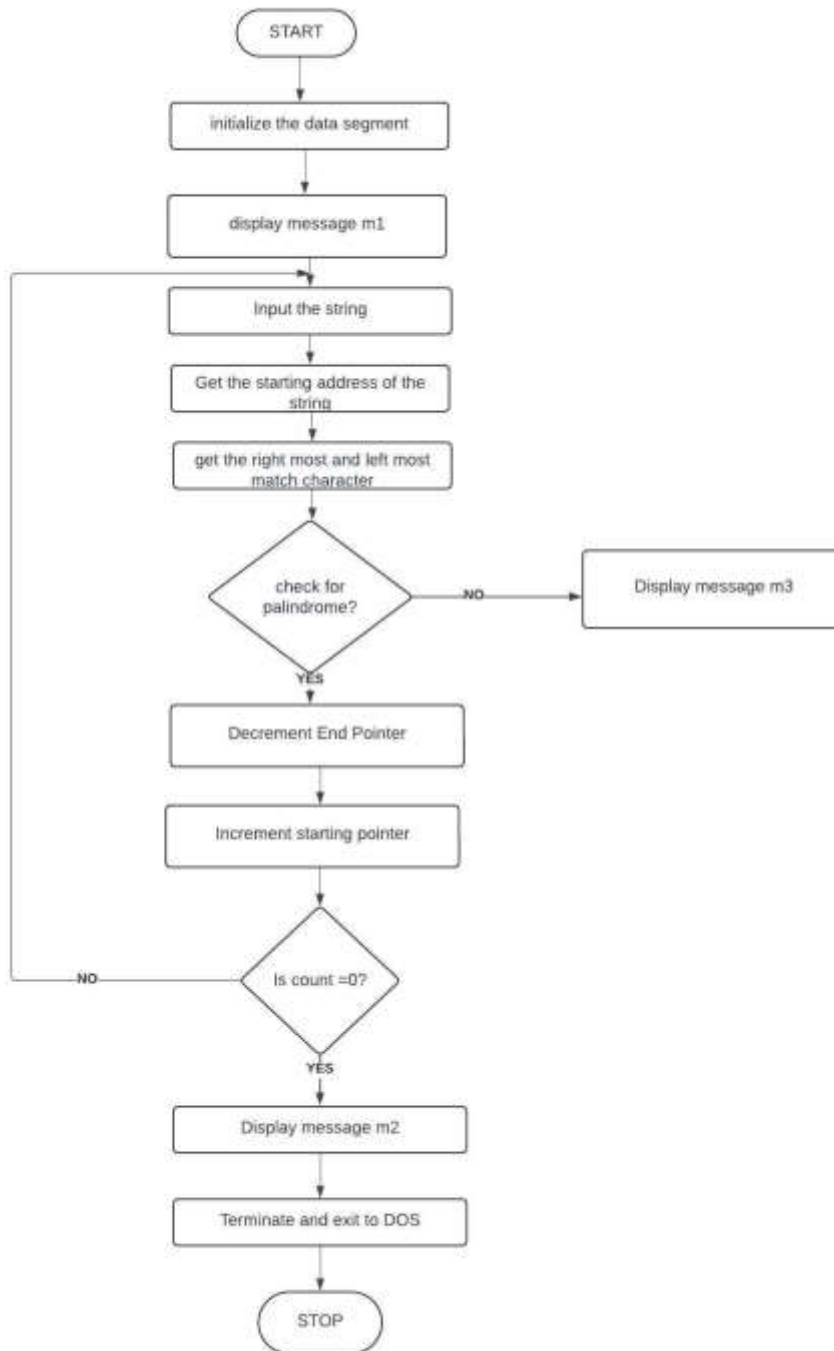
A palindrome string is a string when read in a forward or backward direction remains the same. One of the approach to check this is iterate through the string till middle of the string and compare the character from back and forth.

Algorithm:

1. Initialize the data segment.
2. Display the message M1
3. Input the string
4. Get the string address of the string
5. Get the right most character
6. Get the left most character
7. Check for palindrome.
8. If not Goto step 14
9. Decrement the end pointer
10. Increment the starting pointer.
11. Decrement the counter
12. If count not equal to zero go to step 5
13. Display the message m2
14. Display the message m3
15. To terminate the program using DOS interrupt
 - a. Initialize AH with 4ch
 - b. Call interrupt INT 21h
16. Stop



Flowchart:





Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science (AI&DS)

Program:

org 100h

.data

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

m2 db 10,13,'It is a palindrome\$'

m3 db 10,13,'It 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,m3

mov ah,09h

int 21h

JMP L2

L1:lea dx,m2

mov ah,09h

int 21h

JMP L2

L2:mov ah,4ch

int 21h



Output:

emulator screen (80x25 chars)

```
Enter the string:nadam
It is a palindrome
```

emulatoc: palindrome.com

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

	H	L
AX	4C	24
BX	01	45
CX	00	01
DX	01	16
CS	F400	
IP	0204	
SS	0700	
SP	FFF8	
BP	0000	
SI	0002	
DI	0004	
DS	0700	
ES	0700	

F400:0204

Address	Hex	Dec	Comment
F4200	FF	255	RES
F4201	FF	255	RES
F4202	CD	205	=
F4203	21	033	!
F4204	CF	207	!
F4205	00	000	NULL
F4206	00	000	NULL
F4207	00	000	NULL
F4208	00	000	NULL
F4209	00	000	NULL
F420A	00	000	NULL
F420B	00	000	NULL
F420C	00	000	NULL
F420D	00	000	NULL
F420E	00	000	NULL
F420F	00	000	NULL
F4210	00	000	NULL
F4211	00	000	NULL
F4212	00	000	NULL
F4213	00	000	NULL
F4214	00	000	NULL
F4215	00	000	NULL

BIOS DI

Address	Hex	Dec	Comment
INT 021h			
IRET			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
ADD [BX + SI], AL			
...			

original source co...

```
27 pal:mov ah,[buff+si]
28 mov al,[buff+di]
29 cmp al,ah
30 JC L1
31 inc si
32 dec di
33 loop pal
34
35 lea dx,n3
36 mov ah,09h
37 int 21h
38 JMP L2
39 L1:lea dx,n2
40 mov ah,09h
41 int 21h
42 JMP L2
43 L2:mov ah,4ch
44 int 21h
```

Conclusion: We successfully implemented a program to determine whether a given string is a palindrome or not. The program accepts a string input from the user, then iterates through the characters of the string to compare the characters at corresponding positions from the beginning and end of the string. If all characters match, the program concludes that the string is a palindrome; otherwise, it identifies the string as not a palindrome.



Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science (AI&DS)

1. Explain SAR INSTRUCTION

Ans. The SAR instruction in x86 assembly language stands for "Shift Arithmetic Right.

Purpose: SAR is used to perform a right shift operation on binary data, preserving the sign bit. It shifts the bits of the operand to the right by a specified number of positions.

Syntax: SAR destination, count

Operation: SAR shifts the bits of the destination operand to the right by the number of bit positions specified by the count operand. If the count is greater than 1, the vacated bits are filled with copies of the original sign bit (0 for positive numbers, 1 for negative numbers). If the count is 1, the original sign bit is shifted into the carry flag, and the destination operand is shifted right by one position.

Flags: The flags affected by SAR include CF (carry flag), OF (overflow flag), SF (sign flag), ZF (zero flag), and PF (parity flag). CF is set to the value of the last bit shifted out (the least significant bit before shifting). OF is set if the sign of the operand changes due to the shift (i.e., if the sign bit changes). SF is set according to the sign bit of the result. ZF is set if the result is zero. PF is set according to the parity of the result.

Usage: SAR is commonly used in various arithmetic and logical operations where right-shifting with sign extension is required, such as division by powers of two or signed integer division. It is also used in bit manipulation tasks and data encryption algorithms.

2. Explain DAA instruction.

Ans. The DAA instruction, which stands for "Decimal Adjust Accumulator," is used in x86 assembly language to adjust the result of an addition operation in binary-coded decimal (BCD) format to ensure it represents a valid BCD number. Here's a breakdown of its functionality:

Purpose:

- DAA is used specifically after addition operations involving BCD numbers.
- Its purpose is to adjust the result of the addition to ensure that it remains a valid BCD representation.

Operation:

- After an addition operation involving BCD numbers, DAA examines the content of the accumulator (AL register) to determine if any adjustment is necessary.
- If the low 4 bits of AL (bits 0-3) represent a value greater than 9 or if the Auxiliary Carry flag (AF) is set, DAA adds 6 to AL. This corrects any invalid BCD digits in the low nibble, ensuring they remain in the range of 0 to 9.
- If the high 4 bits of AL (bits 4-7) represent a value greater than 9, or if the Carry flag (CF) is set, DAA adds 96 to AL. This corrects any carry-over from the low nibble to the high nibble.
- The final result is a valid BCD representation in AL.

Flags:

- DAA affects the Carry flag (CF), Auxiliary Carry flag (AF), Sign flag (SF), Zero flag (ZF), and Parity flag (PF).
- CF is updated based on the carry-out from the addition.
- AF is set if a carry occurred from bit 3 to bit 4 during the addition.
- SF, ZF, and PF are updated based on the final value in AL after adjustment.

Usage:

- DAA is commonly used in programs that perform arithmetic operations on BCD numbers, such as financial applications or calculations involving quantities that are naturally represented in decimal format.
- It ensures that the result of BCD arithmetic remains valid and can be correctly interpreted when displayed or used in further calculation.

