SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Electronics & Computer Engineering

Course Name:	**Microprocessors and Microcontrollers Laboratory**	**Semester:**	**IV**
Date of Performance:	**01 / 04 / 2024**	**Batch No.:**	**A - 2**
Faculty Name:	**Kirti Sawlani**	**Roll No.:**	**16014022050**
Faculty Sign & Date:		**Grade / Marks:**	**___ / 25**

Experiment No.: 4
Title: Palindrome Check

Aim and Objective of the Experiment:

Aim: Write an 8086 based ALP to check whether a given string is a palindrome or not. Indicate the same with the messages “Yes it’s a palindrome”, “Sorry, not a palindrome”.

Objectives:

1. To study string instructions.
2. To study DOS interrupts.

This experiment covers:

1. String instructions.
2. DOS interrupts for taking input from keyboard and displaying strings on screen.

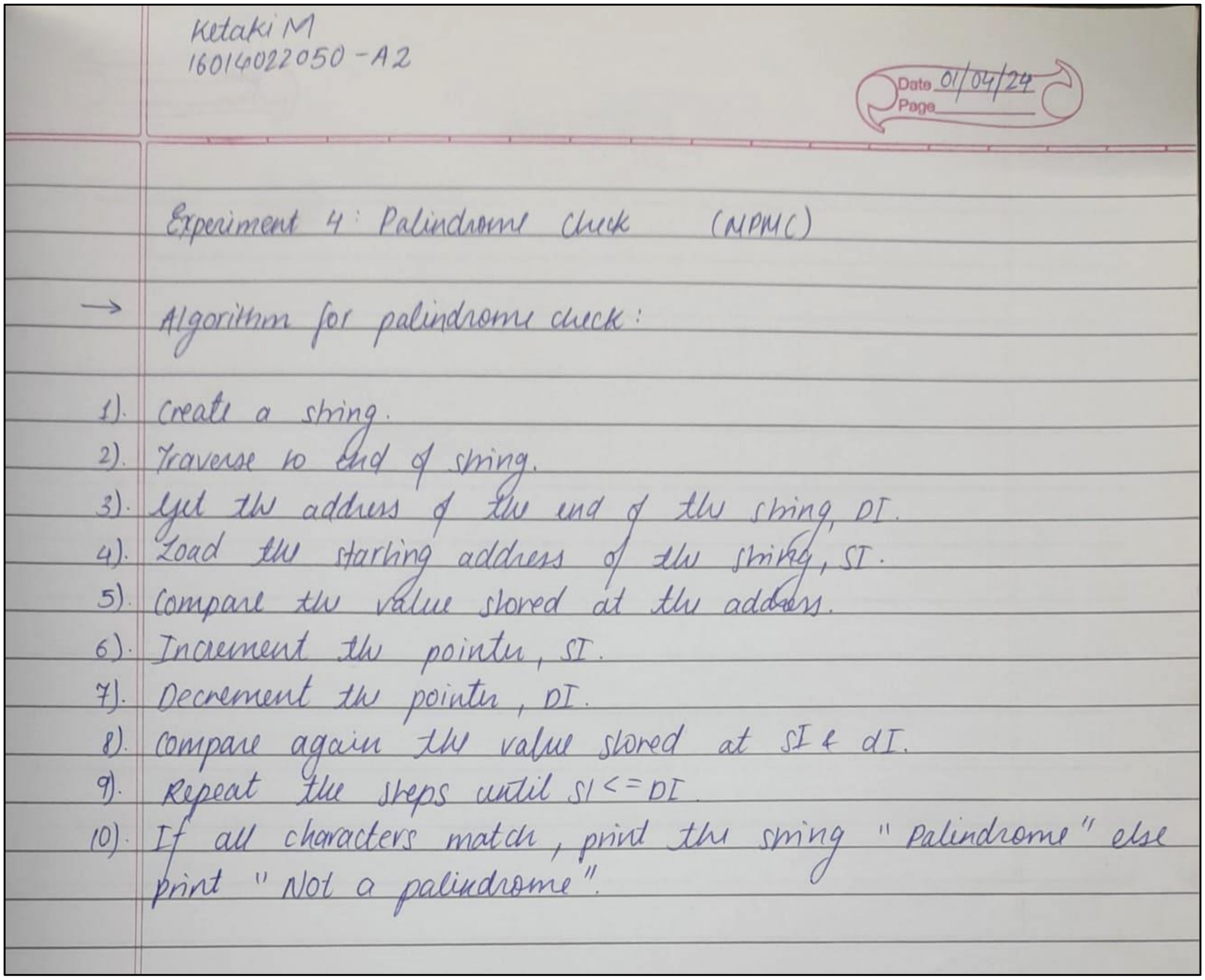
COs to be achieved:

CO2: Develop 8086 based assembly language programs for various applications.

Stepwise-Procedure:

1. Open EMU8086 and write your ASM code in the empty workspace.
2. Click on emulate button and it should open the emulator window.
3. You can run the code using single step execution and monitor the internal registers / flags.

Algorithm / Flowchart:



Ketaki M
16014022050 - A2

Date 01/04/24
Page

Experiment 4: Palindrome check (MPMC)

→ Algorithm for palindrome check:

1). Create a string.
2). Traverse to end of string.
3). Get the address of the end of the string, DI.
4). Load the starting address of the string, SI.
5). Compare the value stored at the address.
6). Increment the pointer, SI.
7). Decrement the pointer, DI.
8). Compare again the value stored at SI & dI.
9). Repeat the steps until SI <= DI.
10). If all characters match, print the string "Palindrome" else print "Not a palindrome".

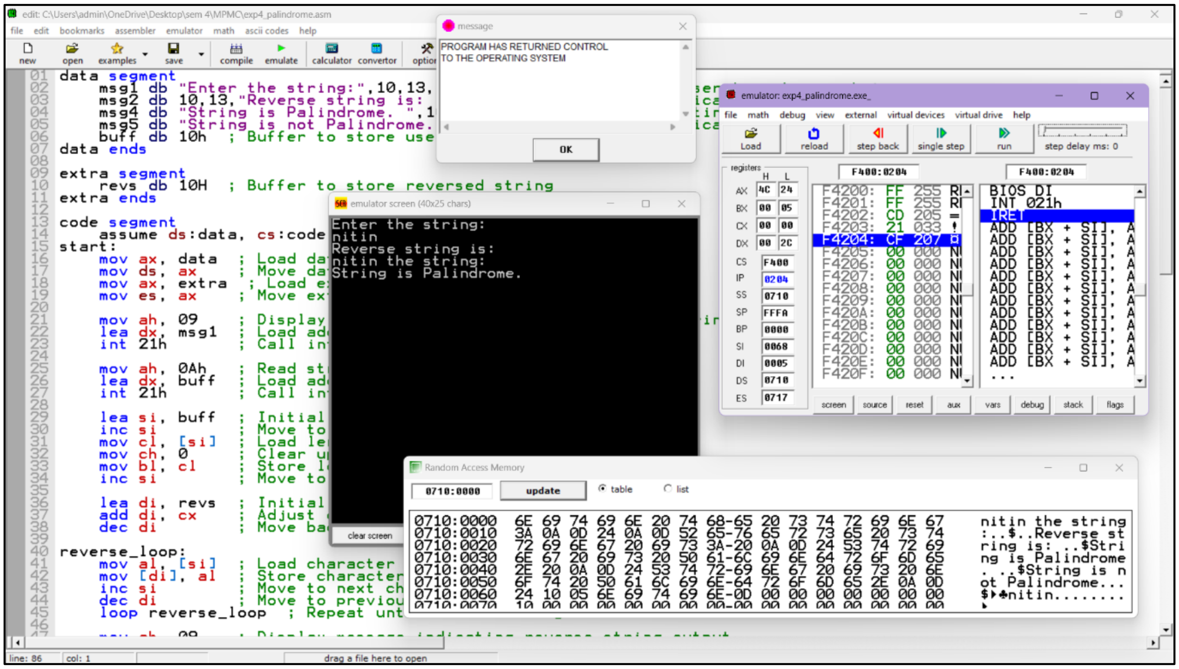
Assembly Language Program:

```
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

exp4_palindrome.exe_ -- emu8086 assembler version: 4.08

[ 01-04-2024  --  23:56:08 ]

===============================================
[LINE]    LOC: MACHINE CODE                      SOURCE
==============================================================

[   1]       :                               data segment
[   2]    0000: 45 6E 74 65 72 20 74 68 65 20 73 74   msg1 db "Enter the string:",10,13,'$'  ; Message
prompting user to enter a string
                72 69 6E 67 3A 0A 0D 24
[   3]    0014: 0A 0D 52 65 76 65 72 73 65 20 73 74   msg2 db 10,13,"Reverse string is: ",10,13,'$'  ;
Message indicating reverse string output
                72 69 6E 67 20 69 73 3A 20 0A 0D 24

[   4]    002C: 53 74 72 69 6E 67 20 69 73 20 50 61   msg4 db "String is Palindrome. ",10,13,'$'  ;
Message indicating string is palindrome
                6C 69 6E 64 72 6F 6D 65 2E 20 0A 0D
                24
[   5]    0045: 53 74 72 69 6E 67 20 69 73 20 6E 6F   msg5 db "String is not Palindrome.",10,13,'$'  ;
Message indicating string is not palindrome
                74 20 50 61 6C 69 6E 64 72 6F 6D 65
                2E 0A 0D 24
[   6]    0061: 10                               buff db 10h  ; Buffer to store user input
[   7]       :                               data ends
[   8]       :
[   9]       :                               extra segment
[  10]    0070: 10                               revs db 10H  ; Buffer to store reversed string
[  11]       :                               extra ends
[  12]       :
[  13]       :                               code segment
[  14]       :                               assume ds:data, cs:code, es:extra
[  15]    0080:                                  start:
[  16]    0080: B8 00 00                             mov ax, data  ; Load data segment into AX register
[  17]    0083: 8E D8                                mov ds, ax    ; Move data segment to DS register
[  18]    0085: B8 07 00                             mov ax, extra  ; Load extra segment into AX register
[  19]    0088: 8E C0                                mov es, ax    ; Move extra segment to ES register
[  20]       :
[  21]    008A: B4 09                                mov ah, 09    ; Display message to prompt user to enter a
string
[  22]    008C: BA 00 00                             lea dx, msg1  ; Load address of msg1 into DX register
[  23]    008F: CD 21                                int 21h       ; Call interrupt to display message
[  24]       :
[  25]    0091: B4 0A                                mov ah, 0Ah   ; Read string from user
[  26]    0093: BA 61 00                 lea dx, buff  ; Load address of buff into DX register
[  27]    0096: CD 21                    int 21h       ; Call interrupt to read string
[  28]        :
[  29]    0098: BE 61 00                 lea si, buff  ; Initialize source index to point to buff
[  30]    009B: 46                       inc si        ; Move to the length byte
[  31]    009C: 8A 0C                    mov cl, [si]  ; Load length of the string
[  32]    009E: B5 00                    mov ch, 0     ; Clear upper byte of CX
[  33]    00A0: 8A D9                    mov bl, cl    ; Store length for later use
[  34]    00A2: 46                       inc si        ; Move to the start of the string
[  35]        :
[  36]    00A3: BF 00 00                 lea di, revs  ; Initialize destination index to point to revs
[  37]    00A6: 03 F9                    add di, cx    ; Adjust destination index to point to the end of revs
[  38]    00A8: 4F                       dec di        ; Move back one byte to ensure correct storage
[  39]        :
[  40]    00A9:                        reverse_loop:
[  41]    00A9: 8A 04                    mov al, [si]  ; Load character from source
[  42]    00AB: 88 05                    mov [di], al  ; Store character to destination
[  43]    00AD: 46                       inc si        ; Move to next character in source
[  44]    00AE: 4F                       dec di        ; Move to previous position in destination
[  45]    00AF: E2 F8                    loop reverse_loop  ; Repeat until entire string is reversed
[  46]        :
[  47]    00B1: B4 09                    mov ah, 09    ; Display message indicating reverse string output
[  48]    00B3: BA 14 00                 lea dx, msg2  ; Load address of msg2 into DX register
[  49]    00B6: CD 21                    int 21h       ; Call interrupt to display message
[  50]        :
[  51]    00B8: B4 09                    mov ah, 09    ; Display reversed string
[  52]    00BA: BA 00 00                 lea dx, revs  ; Load address of revs into DX register
[  53]    00BD: CD 21                    int 21h       ; Call interrupt to display reversed string
[  54]        :
[  55]    00BF: BE 61 00                 mov si, offset buff  ; Reset source index to point to the start of buff
[  56]    00C2: BF 00 00                 lea di, revs         ; Reset destination index to point to the start of revs
[  57]    00C5: 8A CB                    mov cl, bl           ; Load string length for comparison
[  58]    00C7: B5 00                    mov ch, 0
[  59]        :
[  60]    00C9: 46                       inc si               ; Move to the start of the string
[  61]    00CA: 46                       inc si               ; Move to the length byte
[  62]    00CB:                          compare_loop:
[  63]    00CB: 8A 04                        mov al,[si]    ; Load character from input string
[  64]    00CD: 38 05                        cmp [di],al    ; Compare with corresponding character in
reversed string
[  65]    00CF: 75 0D                      jne not_palindrome  ; If not equal, jump to not_palindrome
[  66]    00D1: 46                       inc si          ; Move to next character in input string
[  67]    00D2: 47                       inc di          ; Move to next character in reversed string
[  68]    00D3: E2 F6                     loop compare_loop  ; Repeat until all characters compared
[  69]         :
[  70]    00D5: B4 09                          mov ah, 09     ; Display message indicating string is
palindrome
[  71]    00D7: BA 2C 00                     lea dx, msg4  ; Load address of msg4 into DX register
[  72]    00DA: CD 21                        int 21h        ; Call interrupt to display message
[  73]    00DC: EB 07                       jmp exit_program  ; Jump to exit_program
[  74]         :
[  75]    00DE:                          not_palindrome:
[  76]    00DE: B4 09                        mov ah, 09    ; Display message indicating string is not
palindrome
[  77]    00E0: BA 45 00                     lea dx, msg5  ; Load address of msg5 into DX register
[  78]    00E3: CD 21                       int 21h        ; Call interrupt to display message
[  79]         :
[  80]    00E5:                          exit_program:
[  81]    00E5: B4 4C                       mov ah, 4Ch   ; Exit program
[  82]    00E7: CD 21                       int 21h        ; Call interrupt to terminate program
[  83]         :
[  84]         :                       code ends
[  85]         :                       end start
[  86]         :
[  87]         :

===============================================

EXE HEADER - bytes from 0000 to 01FF inclusive.

0000: 4D     -   exe signature (M)
0001: 5A     -   exe signature (Z)
0002: E9     -   bytes on last page (l.byte)
0003: 00     -   bytes on last page (h.byte)
0004: 02     -   512 byte pages in file (l.byte)
0005: 00     -   512 byte pages in file (h.byte)
```

```
0006: 02     -  relocations (l.byte)
0007: 00     -  relocations (h.byte)
0008: 20     -  paragraphs in header (l.byte)
0009: 00     -  paragraphs in header (h.byte)
000A: 00     -  minimum memory (l.byte)
000B: 00     -  minimum memory (h.byte)
000C: FF     -  maximum memory (l.byte)
000D: FF     -  maximum memory (h.byte)
000E: 00     -  SS - stack segment (l.byte)
000F: 00     -  SS - stack segment (h.byte)
0010: 00     -  SP - stack pointer (l.byte)
0011: 00     -  SP - stack pointer (h.byte)
0012: 06     -  check sum (l.byte)
0013: FB     -  check sum (h.byte)
0014: 00     -  IP - instruction pointer (l.byte)
0015: 00     -  IP - instruction pointer (h.byte)
0016: 08     -  CS - code segment (l.byte)
0017: 00     -  CS - code segment (h.byte)
0018: 1E     -  relocation table adress (l.byte)
0019: 00     -  relocation table adress (h.byte)
001A: 00     -  overlay number (l.byte)
001B: 00     -  overlay number (h.byte)
001C: 01     -  signature (l.byte)
001D: 00     -  signature (h.byte)
001E: 01     -  relocation table - offset inside segment (l.byte)
001F: 00     -  relocation table - offset inside segment (h.byte)
0020: 08     -  relocation table - segment anchor (l.byte)
0021: 00     -  relocation table - segment anchor (h.byte)
0022: 06     -  relocation table - offset inside segment (l.byte)
0023: 00     -  relocation table - offset inside segment (h.byte)
0024: 08     -  relocation table - segment anchor (l.byte)
0025: 00     -  relocation table - segment anchor (h.byte)
0026 to 01FF  -   reserved relocation area  (00)


================================================
```

Output Screenshots:

1. Palindrome: Nitin



edit: C:\Users\admin\OneDrive\Desktop\sem 4\MPMC\exp4_palindrome.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor option

```
data segment
     msg1 db "Enter the string:",10,13,
     msg2 db 10,13,"Reverse string is:
     msg4 db "String is Palindrome. ",1
     msg5 db "String is not Palindrome.
     buff db 10h  ; Buffer to store use
data ends

extra segment
     revs db 10H  ; Buffer to store reversed string
extra ends

code segment
     assume ds:data, cs:code
start:
     mov ax, data   ; Load da
     mov ds, ax     ; Move da
     mov ax, extra   ; Load e
     mov es, ax     ; Move ex

     mov ah, 09     ; Display
     lea dx, msg1   ; Load ad
     int 21h        ; Call in

     mov ah, 0Ah    ; Read st
     lea dx, buff   ; Load ad
     int 21h        ; Call in

     lea si, buff   ; Initial
     inc si         ; Move to
     mov cl, [si]   ; Load le
     mov ch, 0      ; Clear u
     mov bl, cl     ; Store l
     inc si         ; Move to

     lea di, revs   ; Initial
     add di, cx     ; Adjust
     dec di         ; Move ba

reverse_loop:
     mov al, [si]   ; Load character
     mov [di], al   ; Store character
     inc si         ; Move to next ch
     dec di         ; Move to previou
     loop reverse_loop  ; Repeat unt
     mov ah, 09     ; Display message indicating reverse string output
```

```
PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM
```

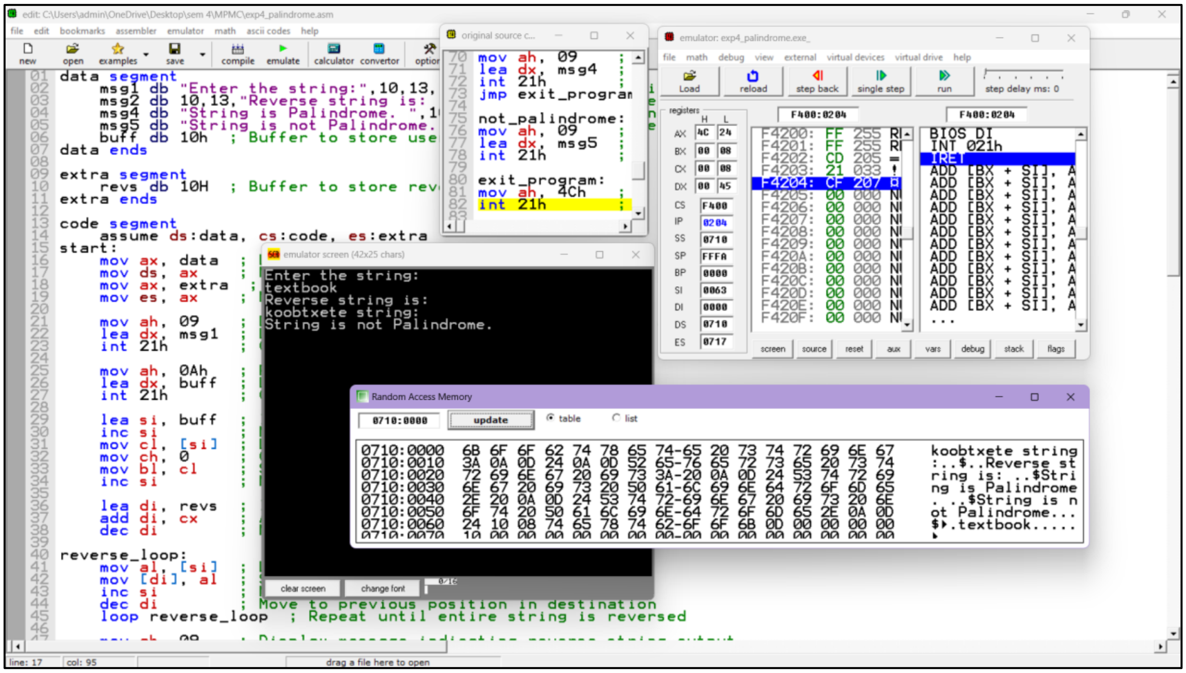
```
Enter the string:
nitin
Reverse string is:
nitin the string:
String is Palindrome.
```

clear screen
emulator: exp4_palindrome.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers screen source reset aux vars debug stack flags

```
     H  L      F400:0204                 F400:0204
AX  4C 24      F4200: FF 255 RI          BIOS DI
BX  00 05      F4201: FF 255 RI          INT 021h
CX  00 00      F4202: CD 205 =           IRET
DX  00 2C      F4203: 21 033 !           ADD [BX + SI], A
CS  F400       F4204: CF 207 ¤           ADD [BX + SI], A
IP  0204       F4205: 00 000 NU          ADD [BX + SI], A
SS  0710       F4206: 00 000 NU          ADD [BX + SI], A
SP  FFFA       F4207: 00 000 NU          ADD [BX + SI], A
BP  0000       F4208: 00 000 NU          ADD [BX + SI], A
SI  0068       F4209: 00 000 NU          ADD [BX + SI], A
DI  0005       F420A: 00 000 NU          ADD [BX + SI], A
DS  0710       F420B: 00 000 NU          ADD [BX + SI], A
ES  0717       F420C: 00 000 NU          ADD [BX + SI], A
               F420D: 00 000 NU          ADD [BX + SI], A
               F420E: 00 000 NU          ADD [BX + SI], A
               F420F: 00 000 NU          ...
```

```
0710:0000

0710:0000  6E 69 74 69 6E 20 74 68-65 20 73 74 72 69 6E 67   nitin the string
0710:0010  3A 0A 0D 24 0A 0D 52 65-76 65 72 73 65 20 73 74   :..$..Reverse st
0710:0020  72 69 6E 67 20 69 73 3A-20 0A 0D 24 53 74 72 69   ring is: ..$Stri
0710:0030  6E 67 20 69 73 20 50 61-6C 69 6E 64 72 6F 6D 65   ng is Palindrome
0710:0040  2E 20 0A 0D 24 53 74 72-69 6E 67 20 69 73 20 6E   . ..$String is n
0710:0050  6F 74 20 50 61 6C 69 6E-64 72 6F 6D 65 2E 0A 0D   ot Palindrome...
0710:0060  24 10 05 6E 69 74 69 6E-0D 00 00 00 00 00 00 00   $▸♣nitin........
0710:0070  10 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ▸
```

2. Not Palindrome: Textbook



edit: C:\Users\admin\OneDrive\Desktop\sem 4\MPMC\exp4_palindrome.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor option

```
data segment
     msg1 db "Enter the string:",10,13,
     msg2 db 10,13,"Reverse string is:
     msg4 db "String is Palindrome. ",1
     msg5 db "String is not Palindrome.
     buff db 10h  ; Buffer to store use
data ends

extra segment
     revs db 10H  ; Buffer to store rev
extra ends

code segment
     assume ds:data, cs:code, es:extra
start:
     mov ax, data   ;
     mov ds, ax     ;
     mov ax, extra   ;
     mov es, ax     ;

     mov ah, 09     ;
     lea dx, msg1   ;
     int 21h        ;

     mov ah, 0Ah    ;
     lea dx, buff   ;
     int 21h        ;

     lea si, buff   ;
     inc si         ;
     mov cl, [si]   ;
     mov ch, 0      ;
     mov bl, cl     ;
     inc si         ;

     lea di, revs   ;
     add di, cx     ;
     dec di         ;

reverse_loop:
     mov al, [si]   ;
     mov [di], al   ;
     inc si         ;
     dec di         ; Move to previous position in destination
     loop reverse_loop  ; Repeat until entire string is reversed
     mov ah, 09     ; Display message indicating reverse string output
```

```
mov ah, 09      ;
lea dx, msg4    ;
int 21h         ;
jmp exit_progran

not_palindrome:
mov ah, 09      ;
lea dx, msg5    ;
int 21h         ;

exit_program:
mov ah, 4Ch     ;
int 21h         ;
```

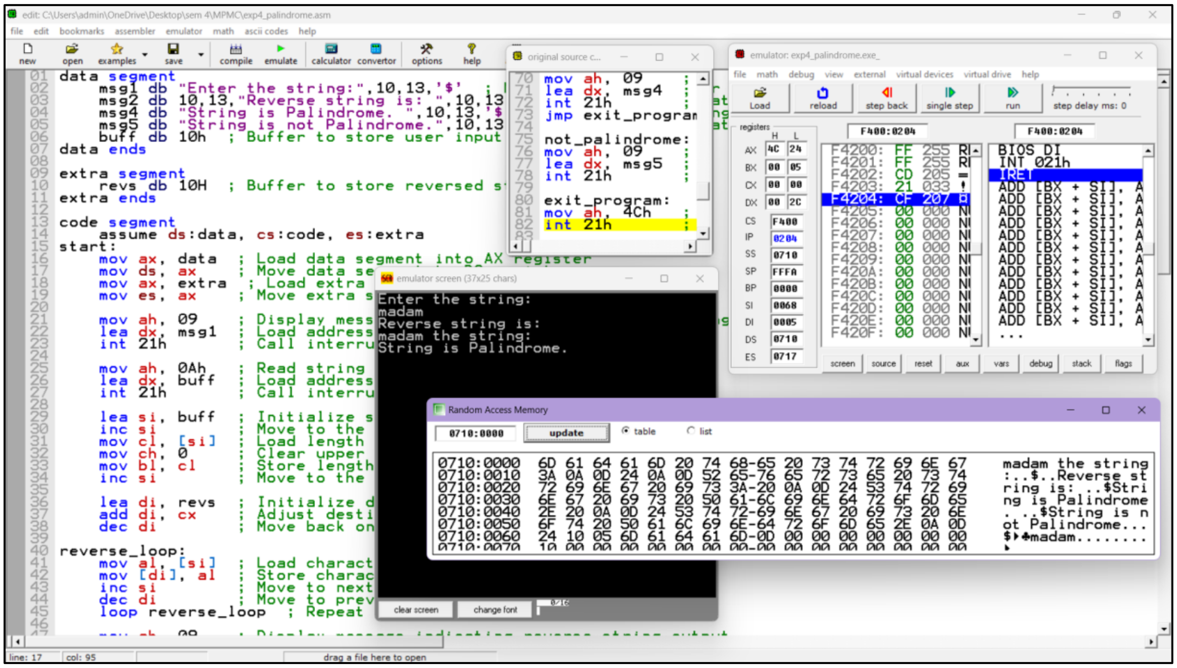
```
Enter the string:
textbook
Reverse string is:
koobtxete string:
String is not Palindrome.
```

clear screen change font 0/16
emulator: exp4_palindrome.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers screen source reset aux vars debug stack flags

```
     H  L      F400:0204                 F400:0204
AX  4C 24      F4200: FF 255 RI          BIOS DI
BX  00 08      F4201: FF 255 RI          INT 021h
CX  00 08      F4202: CD 205 =           IRET
DX  00 45      F4203: 21 033 !           ADD [BX + SI], A
CS  F400       F4204: CF 207 ¤           ADD [BX + SI], A
IP  0204       F4205: 00 000 NU          ADD [BX + SI], A
SS  0710       F4206: 00 000 NU          ADD [BX + SI], A
SP  FFFA       F4207: 00 000 NU          ADD [BX + SI], A
BP  0000       F4208: 00 000 NU          ADD [BX + SI], A
SI  0063       F4209: 00 000 NU          ADD [BX + SI], A
DI  0000       F420A: 00 000 NU          ADD [BX + SI], A
DS  0710       F420B: 00 000 NU          ADD [BX + SI], A
ES  0717       F420C: 00 000 NU          ADD [BX + SI], A
               F420D: 00 000 NU          ADD [BX + SI], A
               F420E: 00 000 NU          ADD [BX + SI], A
               F420F: 00 000 NU          ...
```

```
0710:0000

0710:0000  6B 6F 6F 62 74 78 65 74-65 20 73 74 72 69 6E 67   koobtxete string
0710:0010  3A 0A 0D 24 0A 0D 52 65-76 65 72 73 65 20 73 74   :..$..Reverse st
0710:0020  72 69 6E 67 20 69 73 3A-20 0A 0D 24 53 74 72 69   ring is: ..$Stri
0710:0030  6E 67 20 69 73 20 50 61-6C 69 6E 64 72 6F 6D 65   ng is Palindrome
0710:0040  2E 20 0A 0D 24 53 74 72-69 6E 67 20 69 73 20 6E   . ..$String is n
0710:0050  6F 74 20 50 61 6C 69 6E-64 72 6F 6D 65 2E 0A 0D   ot Palindrome...
0710:0060  24 10 08 74 65 78 74 62-6F 6F 6B 0D 00 00 00 00   $▸.textbook.....
0710:0070  10 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ▸
```

3. Palindrome: Madam


edit: C:\Users\admin\OneDrive\Desktop\sem 4\MPMC\exp4_palindrome.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help

```
data segment
     msg1 db "Enter the string:",10,13,'$'    ;
     msg2 db 10,13,"Reverse string is: ",10,13
     msg4 db "String is Palindrome. ",10,13,'$
     msg5 db "String is not Palindrome.",10,13
     buff db 10h   ; Buffer to store user input
data ends

extra segment
     revs db 10H   ; Buffer to store reversed s
extra ends

code segment
     assume ds:data, cs:code, es:extra
start:
     mov ax, data   ; Load data segment into AX register
     mov ds, ax     ; Move data se
     mov ax, extra   ; Load extra
     mov es, ax     ; Move extra s

     mov ah, 09     ; Display mess
     lea dx, msg1   ; Load address
     int 21h        ; Call interru

     mov ah, 0Ah    ; Read string
     lea dx, buff   ; Load address
     int 21h        ; Call interru

     lea si, buff   ; Initialize s
     inc si         ; Move to the
     mov cl, [si]   ; Load length
     mov ch, 0      ; Clear upper
     mov bl, cl     ; Store length
     inc si         ; Move to the

     lea di, revs   ; Initialize d
     add di, cx     ; Adjust desti
     dec di         ; Move back on

reverse_loop:
     mov al, [si]   ; Load charact
     mov [di], al   ; Store charac
     inc si         ; Move to next
     dec di         ; Move to prev
     loop reverse_loop  ; Repeat
```

```
mov ah, 09       ;
lea dx, msg4     ;
int 21h          ;
jmp exit_program

not_palindrome:
mov ah, 09       ;
lea dx, msg5     ;
int 21h          ;

exit_program:
mov ah, 4Ch      ;
int 21h          ;
```

```
Enter the string:
madam
Reverse string is:
madam the string:
String is Palindrome.
```

clear screen
change font
emulator: exp4_palindrome.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
screen source reset aux vars debug stack flags

```
     H  L
AX  4C 24
BX  00 05
CX  00 00
DX  00 2C
CS  F400
IP  0204
SS  0710
SP  FFFA
BP  0000
SI  0068
DI  0005
DS  0710
ES  0717

F400:0204                F400:0204
F4200: FF 255 RI         BIOS DI
F4201: FF 255 RI         INT 021h
F4202: CD 205 =          IRET
F4203: 21 033 !          ADD [BX + SI], A
F4204: CF 207 ¤          ADD [BX + SI], A
F4205: 00 000 NI         ADD [BX + SI], A
F4206: 00 000 NI         ADD [BX + SI], A
F4207: 00 000 NI         ADD [BX + SI], A
F4208: 00 000 NI         ADD [BX + SI], A
F4209: 00 000 NI         ADD [BX + SI], A
F420A: 00 000 NI         ADD [BX + SI], A
F420B: 00 000 NI         ADD [BX + SI], A
F420C: 00 000 NI         ADD [BX + SI], A
F420D: 00 000 NI         ADD [BX + SI], A
F420E: 00 000 NI         ADD [BX + SI], A
F420F: 00 000 NI         ...
```

```
0710:0000  6D 61 64 61 6D 20 74 68-65 20 73 74 72 69 6E 67   madam the string
0710:0010  3A 0A 0D 24 0A 0D 52 65-76 65 72 73 65 20 73 74   :..$..Reverse st
0710:0020  72 69 6E 67 20 69 73 3A-20 0A 0D 24 53 74 72 69   ring is: ..$Stri
0710:0030  6E 67 20 69 73 20 50 61-6C 69 6E 64 72 6F 6D 65   ng is Palindrome
0710:0040  2E 20 0A 0D 24 53 74 72-69 6E 67 20 69 73 20 6E   . ..$String is n
0710:0050  6F 74 20 50 61 6C 69 6E-64 72 6F 6D 65 2E 0A 0D   ot Palindrome...
0710:0060  24 10 05 6D 61 64 61 6D-0D 00 00 00 00 00 00 00   $▸♣madam........
0710:0070  10 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ▸
```

Post Lab Subjective/Objective type Questions:

1. **Write a short note on REP prefix.**
   The REP (Repeat) prefix is an instruction prefix in x86 assembly language. It is used to repeat certain instructions multiple times based on the value in the CX register or the ECX register (in 32-bit mode) or the RCX register (in 64-bit mode). The REP prefix is commonly used with string manipulation instructions, such as MOVS (move string), CMPS (compare string), SCAS (scan string), and LODS (load string), among others.

   When the REP prefix is used with a string instruction, the string operation is repeated until the CX/ECX/RCX register becomes zero or until the operation's condition is met. For example, REP MOVSB will move bytes from the source to the destination string, and it will continue to do so until CX/ECX/RCX becomes zero.

   The REP prefix allows for efficient handling of repetitive string operations by reducing the number of times the programmer needs to manually specify the instruction. It enhances code readability and conciseness.

2. **List different string instructions.**
   String instructions in x86 assembly language are primarily used for manipulating blocks of data, usually represented as strings of bytes. Here are some commonly used string instructions:

- MOVS (Move String): Copies a block of data from one location to another. It moves a byte, word, or doubleword from the source memory location to the destination memory location. It increments or decrements the source and destination pointers based on the direction flag.

- CMPS (Compare String): Compares the data in two strings. It compares the byte, word, or doubleword at the source memory location with the byte, word, or doubleword at the destination memory location. It increments or decrements the source and destination pointers based on the direction flag.

- SCAS (Scan String): Compares the data in a string with a single byte, word, or doubleword. It compares the byte, word, or doubleword at the source memory location with the specified byte, word, or doubleword in the accumulator register (AL, AX, or EAX). It increments or decrements the source pointer based on the direction flag.

- LODS (Load String): Loads a byte, word, or doubleword from the source memory location into the accumulator register (AL, AX, or EAX). It increments or decrements the source pointer based on the direction flag.

- STOS (Store String): Stores a byte, word, or doubleword from the accumulator register (AL, AX, or EAX) into the destination memory location. It increments or decrements the destination pointer based on the direction flag.

- REP (Repeat): Prefix used with string instructions to repeat the operation specified by the instruction until the CX/ECX/RCX register becomes zero or until the condition for terminating the operation is met.

These string instructions provide powerful capabilities for manipulating strings of data efficiently in x86 assembly language. They are often used in tasks such as string manipulation, searching, sorting, and data processing.

Conclusion:

In conclusion, through the use of assembly language 8086, we've learned an efficient method to determine whether a given string is a palindrome or not. By leveraging string manipulation instructions such as MOVS, CMPS, and the REP prefix, we can compare characters from both ends of the string, ultimately discerning its palindrome status.

Signature of faculty in-charge with Date: