* Binary Search :

```
.MODEL SMALL
; Macro to Display Message

DISPLAY MACRO MSG
        LEA DX, MSG
        MOV AH, 09H
        INT 21H
ENDM

.DATA
LIST DB 01H, 05H, 07H, 10H, 12H, 14H
NUMBER EQU ($ - LIST)

KEY DB 012H

MSG1 DB 0DH, 0AH, "ELEMENT FOUND IN THE
                   LIST... $"

MSG2 DB 0DH, 0AH, " SEARCH FAILED!!
                   ELEMENT NOT FOUND "

.CODE

START : MOV AX, @DATA
        MOV DS, AX
```

displaying msg.

// loads base address to NUMBER

```
MOV  CH, NUMBER-1        ; Highvalue
MOV  CL, 00H             ; lowvalue

AGAIN : MOV  SI, OFFSET LIST   // LEA

        XOR  AX, AX            // clear AX
        CMP  CL, CH            ; ck ch  cl=ch  cl>ch
        JE   NEXT              ① cf=1↓  zf=1↓  cf=0↓
        JNC  FAILED              cl=ch => zf=1

NEXT:   MOV  AL, CL
        ADD  AL, CH
        SHR  AL, 01H           ; Divide By 2
        MOV  BL, AL
        XOR  AH, AH            ; clear AH
        MOV  BP, AX
        MOV  AL, DS : [BP][SI]

        CMP  AL, KEY           ; compare Key
                                 and A[I]

        JE   SUCESS            ; If equal, Display
        JC   JNCLOW              sucess message.

        MOV  CH, BL
        DEC  CH                ; If KEY > A[I]
                                 shift High

        JMP  AGAIN
```

```
INCLOW : MOV CL, BL          ; If KEY<API]
         INC CL                 Shift low
         JMP AGAIN

SUCCESS : DISPLAY MSG1
          JMP FINAL

FAILED :  DISPLAY MSG2         ; JOB OVER
                                 TERMINATE

FINAL :  MOV AH, 4CH
         INT 21H
```

classmate
Date
Page

* Printing the equivalent ASCII value :

```
.model small
.data

msg1    db 0dh, 0ah, "Enter alphanumeric
                      characters $"
res db 0
res     db 02 dup(0)

.code

mov ax, @data
mov ds, ax

lea dx, msg1
call disp

mov ah, 01h
int 21h

mov bl, al
mov cl, 4
shr, al, cl
cmp al, 0ah
JC digit

ADD AL, 07h
```

```
digit : add al, 30h
        mov res, al
        and bl, 0fh
        cmp bl, 0ah
        JC digit1
        add bl, 07h

digit1 : add bl, 30h
         mov res+1, bl

         mov ah, 00h        ; TEXT MODE
         mov al, 03h        ; BIOS
         int 10h

         mov ah, 02h
         mov bh, 00h
         mov dh, 0ch
         mov dl, 26h
         int 10h

mov res+2, '$'
lea dx, res
call disp
ret
disp endp
end
```

LAB-3

Impro

* Pallindrom:

```
.model small

Display macro msg                    // display MSG
        lea DX, msg                  // DX always
        mov AH, 09H
        INT 21H

ENDM                  // end macro

.DATA                 // data segment

msg1 DB 0dbh, 00h, "Enter STRING $"
msg2 db 0dbh, 00h, "Reverse STRING IS $"
msg3 db 0dbh, 00h, " IS A PALLINDROME $"
msg4 db 0dbh, 00h, " NOT A PALLINDROME $"

STRING DB 80H DUP (?)          // memory allocation
REVSTRING DB 80H DUP (?)

.code                 // code segment

START : mov AX, @DATA
        mov DS, AX
        DISPLAY msg1          // for displaying certain msg.

        // mov SI, OFFSET STRING      ; LEA SI, STRING

        XOR CL, CL            // clears CL
```

mov SI, OFFSET NAME

no need of [ ] coz instruction itself is LEA

```
AGAIN :  MOV  AH, 01H        } to take input
         INT  21h            } stored in AL
         CMP  AL, 0DH        // 0DH ascii value
         JE   NEXT                of enter.
         MOV  [SI], AL       // assignment or
         INC  SI                  storage.
         INC  CL             // count of string length
         JMP  AGAIN

NEXT :   MOV [SI] , BYTE PTR '$' // Byte operation
                                 // $ end of string.
         DEC  SI
         MOV  CH, CL         // CH = length //
         MOV  DI, OFFSET RESTRING
                             // LEA DI, RESTRING

BACK :   MOV  AL, [SI]       // last element to AL
         MOV  [DI], AL       // to DI
                             //MOV [DI], [SI] ✗
         DEC  SI             } decrements
         INC  DI             } and increments
         DEC  CH             // no. of times looping

         JNZ  BACK
// MOV [DI] , BYTE PTR '$'
         DISPLAY MSG2        } display msg
         DISPLAY RESTRING    } display value.

LEA SI, STRING
LEA DI, RE-
        -STRING
         MOV  SI, OFFSET STRING
         MOV  DI, OFFSET RESTRING
```

end of string '$'

one by one

```
Aq:   MOV  AL, [SI]      // first element
      cmp  AL, [DI]      // compare

      JNE  FAIL          // if not equal fail.
      INC  SI
      INC  DI
      DEC  CX

      JZ   SUCCESS       // completed all chars.

      JMP  Aq

FAIL :  DISPLAY MSG4
        JMP FINAL

SUCCESS :  DISPLAY MSG3

FINAL :  MOV Ah, 4Ch
         int 21h
         END.
```

★ COMPARE STRINGS :-

```
.model small

DISPLAY MACRO MSG
        LEA DX, MSG
        MOV AH, 09H
        INT 21H

ENDM

.DATA
MSG1 DB 0DH, 0AH, "Enter first string ... $"
MSG2 DB 0DH, 0AH, "Enter second string $"
MSG3 DB 0DH, 0AH, "Length of string 1 $"
MSG4 DB 0DH, 0AH, "Length of string 2 $"
MSG5 DB 0DH, 0AH, "Strings are equal"
MSG6 DB 0DH, 0AH, "Strings are not equal"

STRING1 DB 80H DUP(?)
STRING2 DB 80H DUP(?)

.code

START : MOV AX, @DATA
        MOV DS, AX

        DISPLAY MSG1
        MOV SI, OFFSET STRING1
```

```
CALL READSTR
MOV BL, CL
DISPLAY MSG2
MOV SI, OFFSET STRING2
CALL READSTR
PUSH BX
PUSH CX
DISPLAY MSG3
MOV AL, BL
CALL LEN-DIS
DISPLAY MSG4
MOV AL, CL
CALL LEN-DIS
POP CX
POP BX
CMP CL, BL
JNE FAIL
MOV SI, OFFSET STRING1
MOV DI, OFFSET STRING2
CLD

CHK : MOV AL, [SI]
      CMP AL, [DI]
      JNE FAIL
      INC SI
      INC DI
      DEC CL
      JNZ CHK
      DISPLAY MSG5
      JMP FINAL
```

```
LEN_DIS     PROC NEAR
            XOR AH, AH
            ADD AL, 00H
            AAM
            ADD AX, 3030H
            MOV BH, AL
            MOV DL, AH
            MOV AH, 02H
            INT 21H
            MOV DL, BH
            MOV AH, 02H
            INT 21H
            RET
LEN_DIS     ENDP

READSTR     PROC NEAR
            XOR CL, CL

BACK:       MOV AH, 01H
            INT 21H
            CMP AL, 0DH
            JE FINISH
            MOV [SI], AL
            INC SI
            INC CL
            JMP BACK

FINISH:     MOV [SI], BYTE PTR '$'
            RET

READSTR     ENDP
```

FAIL : DISPLAY MSG6
MOV AH, 4CH
INT 21H

END START

* DISPLAY SYSTEM TIME :-

.model small

DISPLAY MACRO MSG
Lea DX, MSG
MOV AH, 09H
INT 21H

ENDM

.DATA

MSG1 DB 0DH, 0AH, "The Time is $"

.code

CH → hour
CL → min
DH → sec

MOV AX, @DATA
MOV DS, AX

MOV AH, 02CH
INT 21H

/ reads time from system

```
MOV AL, CH
AAM
MOV BX, AX
CALL DISP
MOV DL, ':'
MOV AH, 02H
INT 21H
```

(or) `MOV DL, 20H` ascii ; display character one by one.
[DL register only]

```
MOV AL, CL
AAM
MOV BX, AX
CALL DISP
MOV DL, ':'
MOV AH, 02H
INT 21H

MOV AL, DH
AAM
MOV BX, AX
CALL DISP
MOV AH, 02H
INT 21H

DISP PROC NEAR
MOV DL, BH
ADD DL, 30H
MOV AH, 02H
INT 21H
```

```
MOV DL, BL
ADD DL, 30H
MOV AH, 02H
INT 21H
RET
DISP ENDP
END
```

* CURSOUR :- [9H]

```
.MODEL SMALL

DISP MACRO MSG                    // DISP message
     LEA DX, MSG
     MOV AH, 09H
     INT 21H
ENDM

.DATA

MSG1 DB ODH, OAH, "Enter X-cordinate: $"
MSG2 DB ODH, OAH, "Enter Y-Cooedinate: $"
MSG3 DB
       row db 02 dup(0)
.CODE  col db 02 dup(0)

       MOV AX, @DATA          ; always
       MOV DS, AX

       DISP MSG1
       MOV SI, OFFSET ROW
       CALL READ
       DISP MSG2
       [redacted]
       MOV SI, OFFSET COL
       CALL READ

       MOV SI, OFFSET ROW
       MOV AH, [SI]
       INC SI
       MOV AL, [SI]
```

→ 31 32
→ 01 02
→ 12

Date
Page

```
SUB AX, 3030
AAD

MOV DH, AL        ; stored in AL

MOV SI, OFFSET [redacted] COL
MOV AH, [SI]
[redacted] INC SI
MOV AL, [SI]
SUB AX, 3030
AAD

MOV DL, AL ;      ; cursor pointer values
                    stored in DH, DL
                    x, y.

MOV AH, 00        ; code for setting up the
MOV AL, 03H       ;   cursor & location
INT 10H           ;
MOV AH, 02H       ; default of the setting a
INT 21H           ;   cursor

DISP MSG3
JMP FINAL
```

```
READ PROC NEAR
        MOV CX, 02H          ; Two [redacted] inputs
                             ; to be taken

BACK:   MOV AH, 01H          ; interrupt for
        INT 21H              ; input acceptance
        MOV [SI], AL         ; Keyboard entry from AL
        INC SI
        DEC CX
      JNZ BACK
      RET

READ ENDP                    ; for all procedures ENDP

FINAL:  MOV AH, 01H          ; if entry passed
        INT 21H              ; the program terminates
        MOV AH, 4CH          ; normal
        INT 21H              ; end interrupt.
        END
```

* BTICOUNT :- [10H]

```
.MODEL SMALL
.CODE

MOV  CL, 00
MOV  BH, 00H
MOV  BL, 03H
INT  10H

BACK : MOV  BH, 00H
       MOV  DH, 00H
       MOV  DL, 00H
       MOV  AH, 02H
       INT  10H

       MOV  AL, CL
       ADD  AL, 00H
       AAM
       ADD  AX, 3030H
       MOV  CH, AL
       MOV  DL, AH
       MOV  AH, 02H      ;
       INT  21H          ;

       MOV  DL, CH
       MOV  AH, 02H
       INT  21H
```

```
CALL DELAY
 JNC CL
 XOR AX, AX
 CMP CL, 100D
 JNE BACK
JE LAST


DELAY PROC NEAR
      PUSH AX
      PUSH BX
      PUSH CX
    MOV  CX, 00FFh
Aq1:  NOP

      DEC BX
    JNZ Aq1
     DEC CX
     JNZ Aq
      POP CX
      POP BX
      POP AX
     RET
DELAY ENDP

LAST : MOV AH, 4CH
       INT 21H
       END
```

* BUBBLE SORT :-

```
.model small

display macro msg
  lea dx, msg
  int ah, 09h
  int 21h
endm

.data
n db 5
a db 05, 07, 0h, 03, 06
msg1 db 0dbh 0ah, "1>> SORT IN ASCENDING"
msg2 db 0dbh 0ah, "2>> SORT IN DECENDING"
msg3 db 0dbh 0ah, "3>> EXIT $"
msg4 db 0dbh 0ah, "Enter your choice $"
msg5 db 0dbh 0ah, "Invalid choices"

.code.

mov asc, @data
mov ds, ax
display msg1
display msg2
display msg3
display msg4

mov ah, 01h
int 21h
```

```
sub al, 30h
cmp al, 01h
JE assoet
cmp al, 02h
JE dssort
cmp al, 03h
JE FINAL
display msg 5
JMP FINAL

DSSORT:   mov cl, n
          dec cl

outloop : mov ch, cl
          mov SI, 00

inloop :  mov al, a[SI]
          inc SI
          cmp al, al[SI]
          JC noexchg
          xchg al, a[SI]
          mov al [SI-1], al

noexchg :  dec ch
           JNZ in loop
              dec cl
              jn2 outloop
              JMP FINAL
```

```
dessort :
        mov cl, n
        dec cl

outloop1 : mov ch, cl
           mov SI, 00h

inloop1 :  mov al, a[SI]
           inc SI
           cmp al, a[SI]
           JNC nocxch1
           xchg al, a[SI]
           mov a[SI-1], al

noxch1 :   dec ch
           JNZ inloop1
           dec cl
           JNZ outloop1
           JMP FINAL

FINAL :mov ah, 4ch
       int 21h.

END :
```

* NCR Program :-

```
.model small
.data
n db 4
r db 2
ncr db 0

.data
mov ax, @data
mov ds, ax

mov ax, n
mov bx, r
call ncroute

mov ah, 4ch
int 21h

ncroute proc near

    cmp ax, bx
    JE res1
     cmp bx, 0
     JE resn
     dec ax
      cmp bx, ax
      JE incs
      push ax
      push bx

      call ncroute
```

```
pop bx
pop ax

dec ax
push ax
push bx

call nextroute
pop bx
pop ax
ret

res1:   inc ncs
        ret

incs1 : inc ncs

resn  : add ncs, ax
        ret.
```

* FILE PROGRAM :

```
.model small

DISP MACRO msg
    LEA DX, msg
    MOV AH, 09H
    INT 21H
ENDM

.data
MSG1 DB 0DH 0AH, "Filename for creation"
MSG2 DB 0DH 0AH, "File created successfully"
MSG3 DB 0DH 0AH, "Failed"
MSG4 DB 0DH 0AH, "Filename for deletion"
MSG5 DB 0DH 0AH, "Deletion failed"

FNAME1 DB 10 DUP(0)
FNAME2 DB 10 DUP(0)

.code

    MOV AX, @data
    MOV DS, AX
    DISP MSG1
    MOV SI, 00

BACK1 : MOV AH, 01H
        INT 21H

        CMP AL, 0DH
        JE NEXT1
        MOV FNAME1[SI], AL
```

```
        INC SI
        JMP BACK1

NEXT1 : MOV FNAME1[SI], '$'
        LEA DX, FNAME1
        MOV CX, 00

        MOV ah, 3Ch
        int 21 ch
        JC CREATE
        DISP msg2
        JMP DEL

CREATE: DISP msg3

DEL   : disp msg4
        MOV SI, 00

BACK2 : MOV ah, 01h
        int 21h
        cmp al, 0Dh
        JE NEXT2
        MOV FNAME2[SI], AL
        INC SI
        JMP BACK2
```

```
NEXT2 :  MOV PNAME2[SI] , '$'
         LEA DX, PNAME2

         MOV Ah , 41h
         Int 21h
         JC DFOJL
         DISP msg5
         JMP LAST

DFOJL :  DISP msg6

LAST :   MOV Ah, 4Ch
         int 21h
         END.
```