BINARY SEARCH.

```
.model small
; MACRO TO DISPLAY THE 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 in the
                                        list $"

.code
start: mov AX, @data
       mov ds, ax
       mov ch, number-1   ;  High value
       mov cl, 00H        ;  low value
Again: mov SI, OFFSET LIST
       XOR AX, AX
       CMP CL, CH
       JE NEXT
       JNC FAILED
NEXT: MOV AL, CL
       ADD AL, CH
       SHR AL, 01H        ;  DIVIDE BY2
       MOV BL, AL
       XOR AH, AH         ;  CLEAR AH
       MOV BP, AX
       MOV AL, DS:[BP][SI]
       CMP AL, KEY ;  COMPAR KEY AND A[I]
       JE SUCCESS  ;  IF Equal, display success message
```

```
        JC INCLOW
        MOV CH, BL      ; IF KEY > A[I] SHIFT HIGH
        DEC CH
        JMP AGAIN
INCLOW: MOV CL, BL      ; if key < A[I] SHIFT LOW
        INC CL
        JMP AGAIN
SUCCESS: DISPLAY MSG1
        JMP FINAL
FAILED: DISPLAY MSG2 ;  JOB OVER TERMINATE....
FINAL: MOV AH, 4CH
        INT 21H
END START.
```

Bubble sort.

```
.model small
.data
  n db 5
  a db 05, 07, 04, 03, 06
.code
  mov ax, @data
  mov DS, ax
  mov cl, n
  dec cl
outloop: mov ch, cl
        mov SI, 00h
inloop : mov al, a[SI]
         inc SI
         cmp al, a[SI]
        JC  noexch
      xchg al, a[SI]
       mov a[SI-1], al
noexch : Dec ch
         Jnc inloop
         Dec cl
         JNG outloop
         mov ah, 4ch
         int 21h
    end.
```

Program to find ASCII equivalent.

```
.model small
.data
msg1 db 0dh, 0ah, "Enter alphanumeric character $"
res db 02 dup(00)
.code
mov ax, @data
mov ds, ax
lea dx, msg1
call disp
mov ah, 01h
int 21h
mov bl, al
mov cl, 4
shr al, cl
cmp al, 0ah
jc digit
Add al, 07h
digit : add al, 30h
        mov res, al
        and bl, 0fh
        cmp bl, 0ah
        jc digit1
        add bl, 07h
digit1 : add bl, 30h
        mov res+1, bl
        mov ah, 02h
        mov bh, 00h
        mov dh, 0ch
        mov dl, 28h
        int 10h
mov res+2 '$'
lea dx, res
call disp
```

mov ah, 4ch
int 21h

disp proc near
mov ah, 09h
int 21h
~~ret~~
disp end p
end

checking if a string is a palindrome or not

```
.model small
display macro msg
        lea dx, msg
        mov ah, 09h
        int 21h
endm
.data
msg1 db 0dh, 0ah, "Enter String : $"
msg2 db 0dh, 0ah, "Reverse string : $"
msg3                "Input string is a palindrome $"
msg4                "Input string is not a palindrome $"
string db 80h 0ah Dup (?)
Rstring db 80h Dup (?)
.code
Start : mov ax, @data
        mov ds, ax
        display msg1
        ; take the string from keyboard character by character.
        mov SI, offset string
        XOR cl, cl
Again : mov ah, 01h
        int 21h
        cmp al, 0dh
        je next
        mov [SI], Al
        inc SI
        inc CL
        jmp again
NEXT : mov [SI], Byte ptr '$'
        ; string input over.....
        dec SI
        mov ch, cl
```

```
        ; Reverse the string and store in Rstring mov di,
          offset rstring
Back :  mov al, [SI]
        mov [DI], al
        dec SI
        inc DI
        dec ch
        jnz Back
        mov [DI], byte ptr '$'
        Display msg 2
        Display Rstring
        mov SI, offset string
AG :    mov Al, [SI]
        cmp Al, [DI]
        jnz fail
        inc SI
        inc DI
        dec cx
        jz success
        jmp AG
FAIL :  Display msg 4
        jmp final
SUCCESS: Display msg 3
FINAL : mov ah, 4ch
        int 21h
END.
```

Program to compare two strings and see if they are equal or not.

```
.model small
display macro msg
        lea dx, msg
        mov ah, 09h
        int 21h
endm
.data
 msg1 dB 0DH, 0AH "Enter first string :! $"
 msg2 dB 0AH, 0AH "Enter second string ; $"
 msg3 dB 0DH, 0AH "Length of first string : $"
 msg4 dB 0AH, 0AH "Length of second string : $"
 msg5 dB 0DH 0AH "..... string are equal ..... $"
 msg6 dB 0AH, 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 ; store the length of string
       display msg2
       mov SI, offset string2
call readstr
push bx
push cx
display msg3
mov al, bl
call hex_dis
display msg4
mov al, cl
```

```
call len_dis
POP CX
POP bx
cmp cl, dl      ; Compare the length
JNE fail        ; If length are equal, process next statement
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 al, ah
mov Ah, 02H
int 21h
mov dl, bh
mov ah, 02h
int 21h
RET
lendis end P
Readstr proc near
        XOR cl, cl
Back:   mov ah, 01h
```

```
int 21h
cmp AL, DH
JE finish
mov [SI], al
inc SI
inc CL
jmp back
finish : mov [SI], byte ptr '$'
         ret
register endp
fail : display msg 6
final : mov ah, 4ch
        int 21h
end start
```

Display the system time

```
.model small
.code
mov ah, 2ch
int 21h
mov al, ch
aam
mov bx, ax
call disp
mov dl, 20h
mov ah, 02h
int 21h
mov al, cl
aam
mov BX, AX
call disp
mov dl, 20h
mov ah, 02h
int 21h
mov al, dh
aam
mov bx, ax
call disp
mov ah, 4ch
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.
```

Stimulate Decimal up-counter. [0-99]

```
.MODEL SMALL
.CODE
   MOV CL,00

   MOV AH,00H
   MOV AL,03H
   INT 10H
BACK: MOV BH,00H
      MOV DH,00H
      MOV DL,00H
      MOV DL,00H
      MOV AH,02H
      INT 10H
      MOV AL,CL
      ADD AL,00H
      AAM                    ; 0000
      ADD AX,3030H           ; 3030
      MOV CH,AL
      MOV DL,AH
      MOV AH,02H
      INT 21H
      MOV DL,CH
      MOV AH,02H
      INT 21H
      CALL DELAY
      INC CL
      XOR AX,AX
      CMP CL,100D
 JNE BACK
      JE LAST


      DELAY PROC NEAR
         PUSH AX
```

```
        PUSH BX
        PUSH CX
        MOV CX, 00FFH
AG:     MOV BX, 0FFFH
AG1:    NOP
        DEC BX
        JNZ AG1
        DEC CX
        JNZ AG
        POP CX
POP BX
POP AX
RET
DELAY   ENDP
LAST:   MOV AH, 4CH
INT 21H
END
```

Read
v. Input co-ordinates in BCD & move the cursor to specified location

```
.MODEL SMALL
DISP MACRO MSG
  LEA DX, MSG
  MOV AH, 09 H
  INT 21H                    ; dh 12  mov ah, 01h   0102
ENDM                         ; dl 23  int 21h . AAD- 0012
.DATA
ROW DB 02 DUP(0)
COL DB 02 DUP(0)
MSG1 DB 0DH, 0AH, "ENTER X-coordinates : $ "
msg2 DB 0DH, 0ah, " enter y-coordinates : $ "
msg3 DB 0dh, 0ah, " cursor displayed at the correct coordinates $"
.code
  MOV ax, @data
  MOV ds, ax
  disp msg1
  MOV SI, offset row
  call read
  disp msg2
  MOV si, offset col
  call read
  MOV Si, offset row
  MOV ah, [SI]
  INC SI
  MOV AL, [SI]
  SUB AX, 3030H
  AAD
  MOV DH, AL ; row
  MOV SI, OFFSET COL
  MOV AH, [SI]
```

AAD
MOV AL, [SI]
SUB AX, 3030H

```
INC SI
MOV AL, [SI]
SUB AX, 3030H
AAD
MOV DL, AL ; col

MOV AH, 00
MOV AL, 03H
INT 10H
MOV AH, 02H
INT 10H
DISP msg3
JMP FINAL

READ PROC NEAR
MOV CX, 02H
BACK: MOV AH, 01H
INT 21H
MOV [SI], AL
INC SI
DEC CX
JNZ BACK
RET
READ ENDP
FINAL: MOV AH, 01H
INT 21H
MOV AH, 4CH
INT 21H
END
```

Creation and deletion of file.

```
.MODEL SMALL
DISP MACRO MSG
  LEA DX, MSG
  MOV AH, 09H
  INT 21H
ENDM
.DATA
MSG1 DB 0DH, 0AH, "ENTER THE FILE NAME FOR CREATION :-$"
MSG2 DB 0DH, 0AH, 'FILE CREATED SUCCESSFULLY $"
MSG3 DB 0DH, 0AH, " FILE CREATION FAILED $"
MSG4 DB 0DH, 0AH, " ENTER THE FILE NAME FOR DELETION:-$"
MSG5 DB 0DH, 0AH, " FILE DELETED SUCCESSFULLY $"
MSG6 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 21H
JC CFAIL
```

```
DISP MSG2
JMP DEL
CFAIL: 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 FNAME2[SI],'$'
LEA DX, FNAME2

 MOV AH, 4EH
INT 21H
JC OFAIL
DISP MSG5
JMP LAST
OFAIL: DISP MSG6
LAST: MOV AH, 4CH
INT 21H
END.
```

^nC_r using recursion:

```
.MODEL SMALL

.data
n dw 4
r dw 2
ncr dw 0

.code
mov ax, @data
mov ds, ax

mov ax, n
mov bx, r
call ncrpro
call disp
jmp final

ncrpro proc near
        cmp ax, bx   ; r=n
        je res1
        cmp bx, 0    ; r=0
        je res1                ; 3C2 + 3C1
        cmp bx, 1    ; r= 1
        je resn
        dec ax       ; r=n-1
        cmp bx, ax
        je incr
       push ax
       push bx
       call ncrpro
pop bx
pop ax
dec bx
```

```
push ax
push bx
call ncrpro
pop bx
pop ax
ret

res1 : inc ncr
ret

incr: inc ncr
scan: add ncr, ax    ; 1+2+3+3 = 6
ret
ncrpro endp

disp proc near
mov bx, ncr
add bx, 3030h
mov dl, bh
mov ah, 02h
int 21h
mov dl, bl
mov ah, 02h
int 21h
ret
disp endp

final: mov ah, 4ch
int 21h
end
```