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, 0AH, 12H, 14H
NUMBER EQU ($-LIST)
KEY DB 0AH
MSG1 DB 0DH, 0AH, "ELEMENT FOUND IN LIST... $"
MSG2 DB 0DH, 0AH, "SEARCH FAILED! ELEMENT NOT FOUND $"

.CODE
START: MOV AX, @DATA
       MOV DS, AX

       MOV CH, NUMBER ; HIGH VALUE..
       MOV CL, 00H    ; LOW VALUE
AGAIN: MOV SI, OFFSET LIST
       XOR AX, AX
       CMP CL, CH
       JB NEXT
       JNC FAILED
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  : ~~If Equal, Display Sc~~ comp Key & A[I]
JE SUCCESS   ; if EQ, Display Success
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
FINAL : MOV AH, 4CH  ; JOB OVER, TEMINATE
        INT 21H

END START
```

BUBBLE SORT WITH MENU DRIVEN

```
.MODEL SMALL
DISPLAY MACRO MSG
LEA DX, MSG
MOV AH, 09H
INT 21H
ENDM
.DATA
LIST DB 02H, 01H, 34H, 0F4H, 09H, 05H
NUMBER EQU $-LIST
MSG1 DB 0DH, 0AH, "1>>SORT IN ASCENDING ORDER $"
MSG 2 DB 0DH, 0AH, "2>>SORT IN DESCENDING ORDER $"
MSG 3 DB 0DH, 0AH, "3>> EXIT $"
MSG 4 DB 0DH, 0AH, "ENTER CHOICE : $"
MSG 5 DB 0DH, 0AH, "INVALID CHOICE ENTERED $"
.CODE
START : MOV AX, @DATA
MOV DS, AX
LEA SI, LIST
MOV CH, NUMBER 2
DISPLAY MSG 1
DISPLAY MSG 2
DISPLAY MSG 3
DISPLAY MSG 4
MOV AH, 01H
INT 21H
SUB AL, 30H
CMP AL, 01H
JE ASCSORT
CMP AL, 02H
JE DESSORT
CMP AL, 03H
```

```
JE FINAL
DISPLAY MSG5
JMP FINAL
ASCORT : MOV BL, 00H
AGAIN : MOV SI, OFFSET LIST
MOV CL, 00H
MOV BH, CH
SUB BH, BL
NPASS : CMP CL, BH
JNC NEXT
MOV AL, [SI]
MOV BP, 01H
CMP AL, DS: [BP][SI]
JC NOPE
XCHG AL, [SI+1]
XCHG [SI], AL
NOPE : INC CL
INC SI
JMP NPASS
NEXT : INC BL
CMP BL, CH
JC AGAIN
JMP FINAL
DESCORT : MOV BL, 00H
AGAIN I: MOV SI, OFFSET LIST
MOV CL, 00H
MOV BH, CH
SUB BH, BL
NPASS1 : CMP CL, BH
JNC NEXT
MOV AL, [SI]
MOV BP, 01H
CMP AL, DS: [BP][SI]
```

```
JNC_NOPE1
XCHG AL, [SI+1]
XCHG [SI], AL
-NOPE1: INC CL
INC SI
  JMP NPASS1
  NEXT1 : INC BL
  CMP BL, CH
   JC AGAIN1
FINAL:  MOV AH, 4CH
  INT 21H
END START
```

ASCII

```
.model small
.data
msg1 db 0dh, 0ah, "Enter alphanumeric charac: $"
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                ;   '3' = 3A = 3341
shr al, cl                   AL = 3A   AL = 03   BL = 3A
cmp al, 0ah              ;   AL = 3A    3933      0F
jc digit                 ;                        0A
add al, 07h              ;                        341
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
mov al, 03h ; Text mode
int 10h
mov ah, 02h ; set cursor pos
```

```
mov bh, 00h ; page numbers
mov dh, 0ch
mov dl, 28h
int 10h

mov ses+2, 'H'
lea dx, ses
call disp
mov ah, 4ch
int 21h

disp proc near
mov ah, 09h
int 21h
ret
disp endp
end
```

PALLINDROME

```
.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, "Rev string : $"
msg3 db 0DH, 0AH, "It is a pallindrome $"
msg4 db 0DH, 0AH, "It is not a pallindrome $"
string db 80H dup(?)
rstring db 80H dup(?)

.code
start : mov ax, @data
        mov ds, ax
        display msg1
        ;Take string from Keyboard char by char
        mov si, offset string
        xor cl, cl
again : mov ah, 01h
        int 21h
        cmp al, 0DH ; BMSCE ascii value of enter key
                    ; = 0DH
        jc next ; A1 = 42 BMSCE
        mov [si], al
        inc si
        inc cl
        Jump again
next : mov [si], byte ptr "$"
       ; string input over
       dec si
```

```
mov ch,cl
; rev string is store in rstring
  mov di, offset rstring
back: mov al, [si]
      mov [di], al
      dec si
       inc di
       dec ch
       jnz back
       move: [di], byte ptr "$"
       display msg2
       display rstring
       mov si, offset string
       mov di, offset rstring

ag:    mov al, [si]      ; SMSCE, madam
       cmp al, [di]      ; ECSMS, madam
       jne fail
        inc si
        inc di
        JZ success
        Jmp ag
fail:  display msg4
       Jmp final
success: display msg3
final: mov ah, 4ch
        int 21h
  end.
```

Equal strings & length

```
.model small

.data
    str1    DB  10H DUP(0)
    str2    DB  10H DUP(0)
    len1    DB  0DH
    len2    DB  0DH
    msg1    DB  0DH, 0AH, "Enter first string : $"
    msg2    DB  0DH, 0AH, "Enter second string $"
    msg3    DB  0DH, 0AH, "Strings are equal$"
    msg4    DB  0DH, 0AH, "Strings are not equal$"
    msg5    DB  0DH, 0AH, "Length of first string is $"
    msg6    DB  0DH, 0AH, "Length of second string is $"
    msg7    DB  0DH, 0AH, "Length of string is $"

.code
    MOV AX, @DATA
    MOV DS, AX
    LEA DX, msg1
    MOV AH, 09H
    INT 21H
    MOV SI, 0

BACK1:
    MOV AH, 01H
    INT 21H
    CMP AL, 0DH
    JE NEXT1
    MOV STR1[SI], AL
    INC SI
    INC LEN1
    JMP BACK1
```

```
Next1:
        Lea DX, MSG2
        MOV AH, 09H
        Int 21H
        MOV SI, 00

Back2:
        MOV AH, 01H
        Int 21H
        CMP AL, 0DH
        JE NEXT2
        MOV STR2[SI], AL
        INC SI
        INC Len2
        Jmp Back2

Next2:
        MOV AL, Len1
        CMP AL, Len2
        JNE NOTEQUAL

        MOV SI, 00
        MOV DI, 06
        MOV CL, Len1

Back3:
        MOV AL, Str1[SI]
        CMP AL, Str2[DI]
        JNE NOTEQUAL
        INC SI
        INC DI
        DEC CL
        JNE BACK3
```

```
Lea DX, MSG3
MOV AH, 09H
Int 21H
MOV DL, Len1
ADD DL, 30H
MOV AH, 02H
Int 21H
JMP LAST

NOTEQUAL:
        Lea DX, MSG4
        MOV AH, 09H
        Int 21H
        Lea DX, MSG5
        MOV AH, 09H
        Int 21H
        MOV DL, Len1
        ADD DL, 30H
        MOV AH, 02H
        Int 21H

        Lea DX, MSG6
        MOV AH, 09H
        Int 21H
        MOV DL, Len2
        ADD DL, 30H
        MOV AH, 02H
        Int 21H

Last:
        MOV AH, 4CH
        Int 21H
        End
```

NCR

```
.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

ncrpro proc near
        cmp ax, bx      ; r=n
        je resl
        cmp bx, 0       ; r=0
        je resl
        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 ncs proc
pop bx
pop ax
ret

resl : inc ncs
ret
ncs : inc ncs
non : add ncs, ax ;
ret
ncs proc endp .

disp proc ncas
    mov bx, ncs
    add be, 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
```

System time

```
.model small
display macro msg
lea dx, msg
mov AH, 09H
Int 21H
Endm
.data
Time DB 20 H DUP (?)
msg 1 0DH, 0AH, "Current Time : $"

.code
mov AX, @data
mov DS, AX
; clear screen
mov AH, 00H
mov AL, 03H
Int 10h
AG:     mov BH, 00h
        mov dh, 01h
        mov dl, 01h
        mov ah, 02h
        Int 10h
        lea SI, Time
        mov ah, 2Ch
        Int 21h

mov AL, ch
AAM
Add AX, 3030h
MOV [SI], Ah
Inc SI
```

```
mov [SI], AL
Inc SI
mov [SI], Byte ptr ':'
Inc SI

mov AL, CL
AAM
ADD AX, 3030h
mov [SI], AH
Inc SI
mov [SI], AL
Inc SI
mov [SI], Byte ptr ':'
Inc SI

mov AL, DH
AAM
ADD AX, 3030h
mov [SI], AH
Inc SI
mov [SI], AL
Inc SI
mov [SI], Byte ptr 'H'

Disp msg2
Disp TIME

mov AH, 0BH
Int 21H
cmp AL, 00H
JE AG
mov AH, 4CH
Int 21h
End
```

Decimal up counter

```
.model small
.code
mov cl, 00          ;00
mov Ah, 00h
mov Al, 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                 ;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, 0FFh
AG1: NOP
DEC BX
JNZ AG1
DEC CX
JNZ AG
Pop CX
pop BX
pop AX
Ret
Delay Endp
Last: Move AH, 4ch
Int 21h
End.
```

Cursor to designated coordinates

```
.model small
Disp macro msg
    lea dx, msg
    mov ah, 09h
     Int 21h
Endm
.data
 Row DB 02  Dup(0)
 col DB 02  Dup (0)
 msg 1 DB 0DH, 0AH, "Enter X-co-ordinate: $"
 msg 2 DB 0DH, 0Ah, "Enter Y-co-ordinate: $"
 msg 3 db 0dh, 0ah, "Cursor displayed at cursor
                     co-ordinates $"
.code
 mov AX, @ data
 mov DS, CX
  Disp msg 1
  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
     mov SI , offset col
      mov AH, [SI]
       Inc SI
       mov AL, [SI]
```

```
Sub AX, 3030h
AAD
Mov dl, Al

Mov Ah, 00
mov Al, 03h
int 10h
  mov Ah, 02h
  int 10h
  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
```

File Creak & delete

```
.model small
Disp  macro msg
      lea  dx, msg
      Mov  AH, 09h
      Int 21h
Endm

.data
msg1 DB 0dh, 0ah, "Enter file name to be created $"
 msg2 db 0dh, 0ah, "Creation was successfull $"
msg3 db 0dh, 0ah, "Creation failed $"
msg4 db 0dh, 0ah, "Enter file name to be deleted $"
 msg5 db 0dh, 0ah, "Deletion was successfull $"
 msg6 db 0dh, 0ah, "Deletion failed $"
 fname1 db 10 dup(0)
 fname2 db 10 DUP(0)

.code
mov  Ax, @data
mov  ds, ax

lea  SI, fname1
 disp msg1

L1 : mov AH, 01h
     Int 21h
      Cmp AL, 0dh
      JE  next1
      mov [SI], AL
      Inc SI

 Jmp L1
```

```
Next 1 : mov [SI] , Byte ptr '$'
         Lea dx, fname1
         xor cx, cx
         mov Ah, 3Ch
         int 21h
         JC Cfail
         disp msg2
         Jmp del

Cfail : disp msg 3

del : Lea SI , fname2
      disp msg 4

L2 :  mov AH, 01h
      Int 21h
      Cmp AL, 0Dh
      Je next2
      mov [SI], AL
      Inc SI
    Jmp L2
next2: mov [SI], Byte ptr '$'
       Lea Dx fname2
       mov ah, 41h
       int 21h
       JC dfail
       Disp msg 5
       Jmp final

dfail : Disp msg 6
final : mov ah, 4ch
        Int 21h
        End
```