Ascending order with Address

```
CODE SEGMENT
ASSUME CS:CODE
START:
    MOV AX, 0000H
    MOV BL, AL
    MOV CL, AL
    MOV SI, 1000H
    MOV BL, [SI]
    DEC BL
L3:
    MOV CL, BL
    MOV SI, 2000H
L2:
    MOV AL, [SI]
    CMP AL, [SI+1]
    JLE L1
    XCHG AL, [SI+1]
    MOV [SI], AL
L1:
    INC SI
    LOOP L2
    DEC BL
    JNZ L3
    INT 3
CODE ENDS
END START
```

Descending order without address

```
assume cs:code,ds:data
    data segment
    list dw 03h, 04h, 01h,
05h, 02h
    count equ 05h
data ends
code segment
start:
    mov ax,data
    mov ds,ax
    mov bx,count
    dec bx
back:
    mov cx,bx
    mov si,offset list
again:
    mov ax,[si]
    cmp ax,[si+2]
    jnc go
    xchg ax, [si+2]
    xchg ax, [si]
go:
    inc si
    inc si
    loop again
    dec bx
    jnz back
    hlt
code ends
end start
```

Smallest Number without address

```
assume cs:code,ds:data
data segment
list db
50h,20h,70h,60h,10h,01h,03h
count equ 06h
largest db 01h dup(?)
data ends
code segment
start:mov ax,data
      mov ds,ax
      mov si,offset list
      mov cl,count
      mov al,[si]
again:cmp al,[si+1]
      jle next
      mov al,[si]
next:inc si
     dec cl
     jnz again
     mov si,offset largest
     mov ah,4ch
     int 21h
code ends
end start
```

Binary to Ascii

```
CODE SEGMENT
ASSUME CS:CODE
START:
      MOV AX, 0000H
      MOV SI, 3000H
      MOV AL, [SI]
      MOV AH, AL
      INC SI
      AND AL, 0FH
      MOV CL, 04H
      SHR AH, CL
      OR AX, 3030H
      MOV [SI], AX
      INT 3
CODE ENDS
END START
```

Binatry to Gray Scale

```
CODE SEGMENT
ASSUME CS:CODE
START:
      MOV SI, 3000H
      MOV AL, [SI]
      MOV BL, AL
      CLC
      RCR AL, 1
      XOR BL, AL
      INC SI
      MOV [SI], BL
      INT 3
CODE ENDS
END START
```

Factorial

```
code segment
assume cs:code
start:
        xor ax,ax
        mov bl,bl
        mov si,2000h
        mov bl,[si]
        mov al,01h
l1:     mul bl
        dec bl
        jnz l1
        mov si,3000h
        mov [si],ax
int 3
code ends
end start
```

Display String

```
data segment
array db 'good morning $'
data ends

code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
lea dx,array
mov ah,09
int 21h
mov ah,4ch
int 21h
code ends
end start
```

Reverse String

```
data segment
        str db 'welcome'
        count equ 07
data ends

code segment
assume cs:code,ds:data
start:
        mov ax,data
        mov ds,ax
        mov cx,count
l1:
        mov bx,[si]
        push bx
        inc si
        loop l1
        mov cx,count
l2:
        pop dx
        mov ah,2
        int 21h
        loop l2
        mov ah,4ch
        int 21h
code ends
end start
```

Palindrome

```
data segment
        msg1 db 'Enter the
string:$'
        msg2 db 'String is
Palindrome $'
        msg3 db 'String is Not
Palindrome $'
        str db 50 dup(0)
data ends
code segment
assume cs:code,ds:data
start:
        mov ax,data
        mov ds,ax
        lea dx,msg1
        mov ah,09h
        int 21h
        lea si,str
        lea di,str
        mov ah,01h
next:
        int 21h
        cmp al,0Dh
        je terminate
        mov [di],al
        inc di
        jmp next
        terminate:
        mov al,'$'
        mov [di],al
dothis:
        dec di
        mov al,[si]
        cmp [di],al
        jne notpalindrome
        inc si
        cmp si,di
        jl dothis
palindrome:
        mov ah,09h
        lea dx,msg2
        int 21h
        jmp xx
notpalindrome:
        mov ah,09h
        lea dx,msg3
        int 21h
xx:
        mov ah,4ch
        int 21h
code ends
end start
```

Password Verifying

```
.model small
.stack 100h
.data
message db 'Enter the
pswd:$'
password db 'vitvellore'
count dw 10
correct db 'Password verified
and correct$'
notcorrect db 'Incorecet
password$'
.code
start:
mov ax,@data
mov ds,ax
mov cx,count
mov bx,offset password
mov dx,offset message
mov ah,09h
int 21h
again:
mov ah,08h
int 21h
cmp al,[bx]
jne error
inc bx
loop again
mov dx,offset correct
mov ah,09h
int 21h
jmp exit
error:
mov dx,offset notcorrect
mov ah,09h
int 21h
exit:
mov ah,4ch
int 21h
end start
```

String Lenght

```
data segment
x db "cse2006"
len1 db ($-x)
y db "microprocessor and
interfacing"
len2 db ($-y)
data ends

code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov cl,len1
mov ch,len2
int 3
code ends
end start
```

String Comparison

```
data segment
        array1 db 'good'
        array2 db 'god'
        cnt db 04h
        str1 db 'strings are equal
$'
        str2 db 'strings are
unequal $'
data ends

code segment
assume
cs:code,ds:data,es:data
start:
        mov ax,data
        mov ds,ax
        mov es,ax
        mov cl,cnt
        lea si,array1
        lea di,array2
        rep cmpsb
        jnz l1
        lea dx,str1
        jmp l2
l1:
        lea dx,str2
l2:
        mov ah,09h
        int 21h
        mov ah,4ch
        int 21h
        code ends
        end start
```

fibonacci series

```
.MODEL SMALL
.STACK 100H

.DATA
   fib1 DW 0000H
   fib2 DW 0001H
   fib3 DW 0000H

.CODE
   MOV AX, @DATA
   MOV DS, AX

   MOV CX, 10H       ; loop 10
times
   MOV BX, 0         ; initialize loop
counter
   MOV AX, fib1
   MOV DX, fib2

FIB_LOOP:
   ADD AX, DX        ; add previous
two terms
   MOV fib3, AX      ; save the
result in fib3
   MOV AX, DX        ; shift the
values
   MOV DX, fib3
   INC BX            ; increment the
loop counter
   CMP BX, CX        ; compare the
counter with the limit
   JB FIB_LOOP       ; jump to the
loop if less than

   MOV AX, 4C00H     ; return
control to the operating system
   INT 21H
END
```