Linear Search

```
data segment
a db 10,11,12,13,14,15
len db $-a
key db 13
msg1 db 'key is found''$'
msg2 db 'key is not found''$'
data ends
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
lea si,a
mov cl,len
mov ch,0

l1:mov al,[si]
cmp key,al
je l2
inc si
loop l1
lea dx,msg2
jmp disp
l2:lea dx,msg1
disp:mov ah,09h
int 21h
jmp exit
exit:mov ah,4ch
int 21h
code ends
end start
```

Binary Search

```
assume cs:code,ds:data
data segment
a db 1,3,5,6,8,9
len dw $-a
key db 9
msg1 db 'key found at position $'
msg2 db 'key not found''$'
data ends
code segment
start: mov ax,data
mov ds,ax
mov bx,0
mov cl,key
mov dx,len
back:   cmp bx,dx
        ja notfound
        mov si,bx
        add si,dx
        shr si,1
        cmp a[si],cl
        jle l0
mov dx,si
dec dx
jmp back
l0: je found
mov bx,si
inc bx
jmp back
found:lea dx,msg1
      mov ah,09h
      int 21h
      mov cx,si
      mov dl,cl
      inc dl
      add dl,30h
      mov ah,06h
      int 21h
      jmp exit
notfound: lea dx,msg2
          mov ah,09h
          int 21h
exit:mov ah,4ch
     int 21h
code ends
end start
```

ASCII in the center of screen

```
CODE SEGMENT
ASSUME CS:CODE
START :
MOV AH,1H
INT 21H
PUSH AX
MOV AX,0003H
INT 10h
MOV DH,24
MOV DL,40
MOV AH,2
MOV BH,0
INT 10H
POP AX
mov ah,0
mov bl,1
mul bl
AAM
MOV BX,AX
MOV DL,BH
ADD DL,30H
MOV AH,2
INT 21H
MOV DL,BL
ADD DL,30H
MOV AH,2
INT 21H
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

Macro To read characters from module 1 and display a character in module 2

Mac1.asm

```
read macro
mov ah,01h
int 21h
endm
```

mac2.asm

```
write macro
mov ah,02h
int 21h
endm
```

macro.asm

```
data segment
arr db 20 dup(?)
msg1 db "enter the string:",10,13,"$"
msg2 db "the entered string:$",10,13,"$"
data ends

code segment
assume ds:data,cs:code
start: mov ax,data
      mov ds,ax
include c:\masm\mac1.asm
include c:\masm\mac2.asm
      mov cx,0000h
      lea si,arr
      lea dx,msg1
      mov ah,09h
      int 21h
    r:read
      mov [si],al
      inc si
      inc cx
      cmp al,0dh
      jne r
      lea dx,msg2
      mov ah,09h
      int 21h
      lea si,arr
    w:mov dl,[si]
      write
      inc si
      dec cx
      cmp cx,0
      jne w
      mov ah,4ch
      int 21h
      code ends
      end start
```

Palindrome

```
Data Segment
 str1 db 'MAM','$'
 strlen1 dw $-str1
 strrev db 20 dup(' ')
 str_palin db 'String is Palindrome.','$'
 str_not_palin db 'String is not Palindrome.','$'
Data Ends

Code Segment
 Assume cs:code, ds:data

 Begin:
  mov ax, data
  mov ds, ax
  mov es, ax
  mov cx, strlen1
  add cx, -2

  lea si, str1
  lea di, strrev

  add si, strlen1
  add si, -2
  L1:
    mov al, [si]
    mov [di], al
    dec si
    inc di
    loop L1
    mov al, [si]
    mov [di], al
    inc di
    mov dl, '$'
    mov [di], dl
    mov cx, strlen1

  Palin_Check:
    lea si, str1
    lea di, strrev
    repe cmpsb
    jne Not_Palin
```

```
   Palin:
     mov ah, 09h
     lea dx, str_palin
     int 21h
     jmp Exit

   Not_Palin:
     mov ah, 09h
     lea dx, str_not_palin
     int 21h

   Exit:
     mov ax, 4c00h
     int 21h
code Ends
end Begin
```

String equivalent

f1.asm

```
eread macro
mov ah,1
int 21h
endm
```

str.asm

```
include f1.asm
data segment
m1 db 'enter the string',10,13,'$'
s1 db 10 dup(?)
s2 db 10 dup(?)
m2 db 10,13, 'string are equal $'
m3 db 10,13, 'string are not equal $'
m4 db 10,13, 'length of string 1 $'
m5 db 10,13, 'length of string 2 $'
len1 dw 0
len2 dw 0
data ends
code segment
assume cs:code,ds:data
start:mov ax,data
mov ds,ax
mov es,ax
mov ah,09h
lea dx,m1
int 21h
lea si,s1
read:eread
cmp al,13
```

```
je l
mov [si],al
inc si
inc len1
jmp read
l:lea di,s2
l1:eread
cmp al,13
je l2
mov [di],al
inc di
inc len2
jmp l1
l2:mov cx,len1
cmp cx,len2
jne strn

mov cx,len1
lea si,s1
lea di,s2
cld
repe cmpsb
jne strn
mov ah,09h
lea dx,m2
int 21h
jmp exit
strn:mov ah,09h
lea dx,m3
int 21h
exit:mov ah,09h
lea dx,m4
int 21h
mov dx,len1
add dl,30h
mov ah,2
int 21h
mov ah,09h
lea dx,m5
int 21h
mov dx,len2
add dl,30h
mov ah,2
int 21h
mov ah,4ch
int 21h
code ends
end start
```

Decimal upcounter (00-99) // You can also see another one which is next of this program

1)

```
ASSUME CS: CODE

CODE SEGMENT
START: MOV BX, 00
REPEAT: PUSH BX
      MOV AH, 07H
      MOV AL, 00H
      MOV BH, 0FH
      MOV CX, 00H
      MOV DH, 31H
      MOV DL, 79H
      INT 10H
      MOV AH, 02H
      MOV BH, 00H
      MOV DH, 0CH
      MOV DL, 25H
      INT 10H
      POP BX
      MOV AL, BL
      AAM
      ADD AX, 3030H
      MOV CX, AX
      MOV DL, AH
      MOV AH, 02H
      INT 21H
      MOV DL, CL
      MOV AH, 02H
      INT 21H
      CALL DELAY
      INC BL
      CMP BL, 100
      JNE REPEAT
      MOV AH, 0BH
      INT 21H

     CMP AL, 00H
     JE START


     MOV AH, 4CH
     INT 21H
DELAY PROC NEAR
PUSH CX
PUSH DX
MOV DX, 0FFFH
B2: MOV CX, 0FFFFH
B1: LOOP B1
```

```
DEC DX
JNZ B2
POP DX
POP CX
RET
DELAY ENDP
 CODE ENDS
       END START
```

2)

```
assume cs:code
start:mov al,00
mov bl,00
mov dh,20
mov dl,50
mov ah,2
mov bh,0
int 10h
mov dl,30h
mov ah,02h
int 21h
mov dl,30h
mov ah,02h
int 21h
call delay
call delay
mov al,00
mov bl,00
mov cx,99
l:mov dh,20
mov dl,50
mov ah,2
mov bh,0
int 10h
mov al,bl
add al,1
daa
mov bl,al
push ax
push cx
mov cl,4
str al,cl
add al,30h
mov dl,al
mov ah,2
int 21h
pop cx
pop ax
and al,0fh
```

```
add al,30h
mov dl,al
mov ah,2
int 21h
call delay
call delay
dec cx
jnz l
mov ah,4ch
int 21h

delay proc
push cx
push bx
mov cx,0ffffh
dl1: mov bx,0fffh
dl2: dec bx
jnz dl2
loop dl1
pop bx
pop cx
ret
delay endp
code ends
end start
```

To display System time

```
data segment
res db ?
data ends
code segment
assume cs:code,ds:data
start: mov ax,data
    mov ds,ax
    mov ah,2ch
    int 21h
    mov al,ch
    aam
    add ax,3030h
    mov res,al
    mov dl,ah
    mov ah,2
    int 21h
    mov dl,res
    mov ah,2
    int 21h
    mov dl,':'
    mov ah,2
    int 21h
    mov al,cl
    aam
    add ax,3030h
    mov res,al
    mov dl,ah
    mov ah,2
    int 21h
    mov dl,res
    mov ah,2
    int 21h
    mov ah,4ch
    int 21h
    code ends
    end start
```

Hardware Programs

BCD updown counter

```
data segment
count db 00
pa equ 0dc00h
pb equ 0dc01h
pc equ 0dc02h
cp equ 0dc03h
data ends
code segment
assume cs:code,ds:data
start: mov ax,data
mov ds,ax
mov dx,cp
mov al,82h
out dx,al
mov cx,10
mov al,00
l:mov dx,pa
out dx,al
call delay
inc al
loop l
mov cx,9
mov al,8
l1:mov dx,pa
out dx,al
dec al
call delay
loop l1
mov ah,4ch
int 21h

delay proc
push cx
push bx
mov cx,0ffffh
dl1:mov cx,4fffh
dl2:dec bx
jnz dl2
loop dl1
pop bx
pop cx
ret
delay endp
code ends
end start
```

Ring counter

```
data segment
pa equ 0dc00h
pb equ 0dc01h
pc equ 0dc02h
cp equ 0dc03h
data ends
code segment
assume cs:code,ds:data
start: mov ax,data
mov ds,ax
mov dx,cp
mov al,82h
out dx,al
mov cx,10
mov al,30h
l:mov dx,pa
out dx,al
call delay
ror al,1
loop l
mov ah,4ch
int 21h

delay proc
push cx
push bx
mov cx,0ffffh
dl1:mov cx,4fffh
dl2:dec bx
jnz dl2
loop dl1
pop bx
pop cx
ret
delay endp
code ends
end start
```

Johnson counter

```
data segment
pa equ 0dc00h
pb equ 0dc01h
pc equ 0dc02h
cp equ 0dc03h
data ends
code segment
assume cs:code,ds:data
start: mov ax,data
mov ds,ax
mov dx,cp
mov al,82h
out dx,al
mov cx,10
mov al,00h
l:mov dx,pa
out dx,al
call delay
ror al,1
xor al,80h
loop l
mov ah,4ch
int 21h

delay proc
push cx
push bx
mov cx,0ffffh
dl1:mov cx,4fffh
dl2:dec bx
jnz dl2
loop dl1
pop bx
pop cx
ret
delay endp
code ends
end start
```

FIRE HELP

```
data segment
pa equ 0DC00h
pb equ 0DC01h
pc equ 0DC02h
cp equ 0DC03h
fire db 86h,88h,0f9h,8eh
help db 8ch,0c7h,86h,89h
b db 0ffh
```

```
data ends
code segment
assume cs:code,ds:data
start:mov ax,data
mov ds,ax
mov dx,cp
mov al,80h
out dx,al
mov bx,8
rl:lea si,fire
mov cx,4
l1 : mov al,[si]
call disp
inc si
loop l1
call delay
call delay
mov cx,04
bl1 : mov al,b
call disp
loop bl1
call delay
call delay
lea si,help
mov cx,4
l2:mov al,[si]
call disp
inc si
loop l2
call delay
call delay
mov cx,4
bl2: mov al,b
call disp
loop bl2
call delay
call delay
dec bx
jnz rl
mov ah,4ch
int 21h

delay proc
push bx
push cx
mov cx,0ffffh
dl1:mov bx,2fffh
dl2:dec bx
jnz dl2
loop dl1
pop cx
pop bx
ret
delay endp

disp proc
push cx
mov cx,8
l:rol al,1
mov dx,pb
out dx,al
push ax
mov al,1
mov dx,pc
out dx,al
mov al,0
out dx,al
pop ax
loop l
pop cx
ret
disp endp
code ends
end start
```

Binary to BCD (7 segment display)

```
data segment
pa equ 0DC00h
pb equ 0DC01h
pc equ 0DC02h
cp equ 0DC03h
a db 0ffh
h db 0c0h,0f9h,0a4h,0b0h,99h,92h,82h,0f8h,80h,90h
data ends
code segment
assume cs:code,ds:data
start:mov ax,data
mov ds,ax
mov dx,cp
mov al,80h
out dx,al
mov ax,a
rl:mov ah,0
mov dl,10
div dl
mov bl,ah
mov bh,0
push ax
mov al,h[bx]
call disp
pop ax
```

```
dec disp
jnz rl
mov ah,4ch
int 21h

disp proc
push cx
mov cx,8
l:rol al,1
mov dx,pb
out dx,al
push ax
mov al,1
mov dx,pc
out dx,al
mov al,0
out dx,al
pop ax
loop l
pop cx
ret
disp endp
code ends
end start
```

Keyboard program

```
data segment
pa equ 0DC00h
pb equ 0DC01h
pc equ 0DC02h
cp equ 0DC03h
msg1 db 'Row number: $'
row db ?
msg2 db 'Column number: $'
col db ?
tab db "0123456789.+-*/%pqr=stuv"
msg3 db 'The key is : $'
data ends
code segment
assume cs:code,ds:data
start: mov ax,data
mov ds,ax
mov es,ax
mov al,90h
mov dx,cp
out dx,al
call scan
mov ah,4ch
int 21h
```

```
scan proc
l:mov al,01h
kl:mov dx,pc
out dx,al
mov dx,pa
push ax
in al,dx
cmp al,0
jne kl1
pop ax
rol al,1
cmp al,08h
je 1
jmp kl
kl1: pop bx
mov di,0
mov si,0
kl2: shr al,1
jc kl3
inc di
jmp kl2
kl3: shr bx,1
jc kl4
inc si
jmp kl3
kl4: mov ah,09h
lea dx,msg1
int 21h
mov dx,si
add dl,30h
mov ah,2
int 21h

mov ah,09
lea dx,msg2
int 21h
mov dx,di
add dl,30h
mov ah,2
int 21h

mov cl,3
shl si,cl
add si,di
mov ah,09h
lea dx,msg3
int 21h
mov dl,tab[si]
mov ah,02h
int 21h
ret
```

```
scan endp

code ends
end start
```

Stepper motor program

```
code segment
assume cs:code
start: mov al,80h
mov dx,0dc03h
out dx,al
mov al,33h
mov cx,100
l: mov dx,0dc02h
out dx,al
ror al,1
call delay
loop l
mov ah,4ch
int 21h

delay proc
push cx
push bx,
mov cx,offffh
dl1:mov bx,0fffh
dl2: dec bx
jnz dl2
loop dl1
pop bx
pop cx
ret
delay endp
code ends
end start
```