16bit add

============

```
assume cs:code,ds:data
data segment
        opr1 dw 1144h
    opr2 dw 4477h
    result dw ?

data ends
code segment
    org 0100h
start:  mov ax,data
    mov ds,ax

    mov ax,opr1
    mov bx,opr2
    add ax,bx
    mov result,ax


        mov ah,4ch
    int 21h
        code ends
end start
```

Case conversion

================

```
ASSUME CS:CODE,DS:data

data SEGMENT

COUNT equ 10h

data ends

CODE SEGMENT

START:MOV AX,data

            MOV DS,AX

            MOV CX,COUNT ; LOOP COUNTER

L1:MOV AH,1 ; INPUT CHARACTER,

            INT 21H ; AL = CHARACTER

            CMP AL,60H

            JNC UPPER

            ADD AL,20H

            JMP SKIP

UPPER:SUB AL,20H ; CONVERT TO UPPER CASE

SKIP:MOV AH,2 ; CHARACTER OUTPUT FUNCTION

            MOV DL,AL ; CHARACTER MUST BE IN DL
```

```
        INT 21H ; DISPLAY THE CHARACTER

        LOOP L1 ; REPEAT LOOP

        MOV Ah,4CH

        INT 21H

CODE ENDS

end start
```

Float Add

=========

```
ASSUME          CS:CODESEG, DS:DATASEG

; ------------------------------------------------------------

DATASEG         SEGMENT                             ; start of data segment

        ORG     00H                  ; directive to assign an offset address for a variable

X       DD      20.4375

        ORG     10H

Y       DD      20.4375

        ORG     20H

SUM     DD      ?

DATASEG         ENDS                        ; end of data segment
```

```
; -------------------------------------------------------------

CODESEG         SEGMENT                         ; start of code segment

start:  MOV     AX,DATASEG      ; load the data segment address
        MOV     DS,AX           ; assign value to DS

        FINIT                   ; initialize 8087 stack
        FLD     X               ; load X into ST(0)
        FLD     Y               ; load Y into ST(0)

        FADD    ST(0),ST(1)     ; ST(0) = X+Y

        FST     SUM             ; store ST(0) in sum

        MOV     AH,4CH          ; setup function-4C of the int21
        INT     21H             ; call BIOS int21 to return to DOS



CODESEG         ENDS                    ; end of code segment
        END START
```

F

loat sub

=========

```
ASSUME      CS:CODESEG, DS:DATASEG

; ------------------------------------------------------------

DATASEG     SEGMENT                     ; start of data segment

      ORG   00H         ; directive to assign an offset address for a variable

X     DD    20.4375

      ORG   10H

Y     DD    0.125

      ORG   20H

SUM   DD    ?

DATASEG     ENDS                  ; end of data segment


; ------------------------------------------------------------


CODESEG     SEGMENT                     ; start of code segment


start:  MOV   AX,DATASEG  ; load the data segment address

      MOV   DS,AX       ; assign value to DS
```

```
        FINIT                   ; initialize 8087 stack

        FLD     Y               ; load X into ST(0)

        FLD     X               ; load Y into ST(0)


        FSUB    ST(0),ST(1)     ; ST(0) = X+Y


        FST     SUM             ; store ST(0) in sum


        MOV     AH,4CH          ; setup function-4C of the int21

        INT     21H             ; call BIOS int21 to return to DOS




CODESEG         ENDS                    ; end of code segment

        END START
```

Largest

=========

```
assume cs:code,ds:data

data segment

            count db 00h

            numbers db 10 dup(0)

            result db 00h

data ends

code segment

       org 1000h

start:  mov ax,data

       mov ds,ax


            mov si,offset numbers

            mov cl,count


carry: mov al,[si]

other: inc si

            dec cl

            jz finish

nonzero:cmp al,[si]

            jc carry

            jmp other

finish: mov si,offset result

            mov [si],al
            mov ah,4ch
        int 21h
            code ends
end start
```

SUM of N

=========

```
assume cs:code,ds:data
data segment
            count db 00h
            numbers db 10 dup(?)
            carry db 00h
            result db 00h
data ends
code segment
        org 1000h
start:  mov ax,data
        mov ds,ax
```

```
            mov cl,00h

            mov ax,0000h

            mov dl,count

            mov si,offset numbers

nozero:add al,[si]

            jnc nocarry

            inc cl

nocarry:inc si

            dec dl

            jnz nozero

            mov si,offset result

            mov [si],al

            mov si,offset carry

            mov [si],cl



            mov ah,4ch

        int 21h

            code ends

end start
```

ODD EVEN

=========

```
assume cs:code,ds:data

data segment

            count db 00h

            numbers db 10 dup(0)

            oddcount db 00h

            evencount db 00h

data ends

code segment

       org 1000h

start:  mov ax,data

       mov ds,ax


            mov si,offset numbers

            mov cl,count

            mov ax,0000h

            mov bl,00h

            mov bh,02h

            mov dl,00h

            inc cl

scanlist:mov al,[si]
```

```
        inc si

        dec cl

        jz store

        div bh

        or ah,00h

        jnz odd

        inc bl

        jmp scanlist

odd:    inc dl

        jmp scanlist

store:  mov si,offset evencount

        mov [si],bl

        mov si,offset oddcount

        mov [si],dl


        mov ah,4ch

    int 21h

        code ends

end start
```

Pass

====

```
assume cs:code,ds:data


data segment
    pass db "392001$"
    mes1 db "Password is correct!$"
    mes2 db "Password is incorrect!$"
    disp db "Password : $"
data ends


code segment
  org 0100H
start:
  MOV AX,data
  MOV DS,AX


  MOV AH,09H
  MOV DX,OFFSET disp
  INT 21H


  MOV BX,OFFSET pass
  MOV DI,0006H
```

```
L1: MOV AH,01H

   INT 21H

   MOV DH,AL

   MOV SI,000FH

  L3:

    MOV CX,9FFFH

    L2: LOOP L2

    DEC SI

    JNZ L3


   MOV AH,2

   MOV DL,08H

   INT 21H


   MOV AH,2

   MOV DL,2AH

   INT 21H


   CMP DH,[BX]

   JNE fail

   INC BX

   DEC DI

   JNZ L1


   CMP CX,0000H
   JNE fail

   MOV AH,2

   MOV DL,0AH

   INT 21H

   MOV AH,09H

   MOV DX,OFFSET mes1

   INT 21H

   JMP exit


fail: MOV AH,2

     MOV DL,0AH

     INT 21H

     MOV AH,09H

     MOV DX,OFFSET mes2

     INT 21H


exit: MOV AH,4CH

     INT 21H


     code ends

end start
```

String

======

```
DATA SEGMENT

MESSAGE DB "THIS IS THE STRING$"

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:MOV AX,DATA

MOV DS,AX

MOV AH,9 ; DOS FUNCTION #9

MOV DX,OFFSET MESSAGE ; OFFSET OF THE STRING

INT 21H ; DISPLAY IT

MOV Ah,4CH

INT 21H

CODE ENDS

END START
```

SYSDATE

=======

```
assume cs:code,ds:data

data segment
```

```
            day db 01 dup(?)

            month db 01 dup(?)

            year db 02 dup(?)

data ends

code segment

       org 0100h

start:  mov ax,data

       mov ds,ax




   ;system date

;INT 21h /AH=2Ah - get system date;

;return:CX= year (1980-2099).DH= month. DL= day.AL= day of week (00h=Sunday)


            mov ah,2ah

       int 21h


mov si,offset day

       mov [si],dl


mov si,offset month

       mov [si],dh


mov si,offset year

       mov [si],cx
```

```
          mov ah,4ch
     int 21h
          code ends
end start
```

SYStime

=======

```
assume cs:code,ds:data
data segment
          hour db 01 dup(?)
          minute db 01 dup(?)
          second db 02 dup(?)
data ends
code segment
     org 0100h
start:  mov ax,data
     mov ds,ax

;  INT 21h/AH=2Ch- get system time;
```

```
;return:CH= hour.CL= minute.DH= second


          mov ah,2ch
     int 21h

mov si,offset hour
     mov [si],ch

mov si,offset minute
     mov [si],cl

mov si,offset second
     mov [si],dh

          mov ah,4ch
     int 21h
          code ends
end start
```