Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería



**ORGANIZACIÓN DE LAS COMPUTADORAS Y LENGUAJE ENSAMBLADOR**

**Practica 10**

**Ejercicios de practica en el lenguaje ensamblador del procesador 8086**

**Docente:** Sanchez Herrera Mauricio Alonso

**Alumno:** Gómez Cárdenas Emmanuel Alberto

**Matricula:** 1261509

Contenido

[DESARROLLO 3](#_Toc58879438)

[Inciso 1 3](#_Toc58879439)

[Inciso 2 3](#_Toc58879440)

[Inciso 4 3](#_Toc58879441)

[Inciso 5 3](#_Toc58879442)

[Inciso 6 3](#_Toc58879443)

[Inciso 7 3](#_Toc58879444)

[Inciso 8 3](#_Toc58879445)

[Inciso 9 4](#_Toc58879446)

[Inciso 10 4](#_Toc58879447)

[Inciso 11 4](#_Toc58879448)

[Inciso 12 4](#_Toc58879449)

[Inciso 13 4](#_Toc58879450)

[Inciso 14 4](#_Toc58879451)

[Inciso 15 5](#_Toc58879452)

[Inciso 16 5](#_Toc58879453)

[Inciso 17 5](#_Toc58879454)

[Inciso 18 5](#_Toc58879455)

[Inciso 2 5](#_Toc58879456)

[Inciso 4 5](#_Toc58879457)

[Inciso 10 5](#_Toc58879458)

[Inciso 14 6](#_Toc58879459)

[Inciso 19 6](#_Toc58879460)

[Inciso 22 6](#_Toc58879461)

[Inciso 23 6](#_Toc58879462)

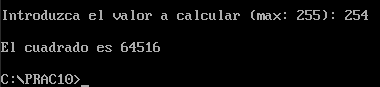
[CONCLUSIONES 6](#_Toc58879463)

DESARROLLO

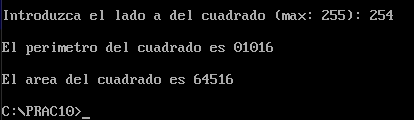
## [Inciso 1](#_INCISO_1)



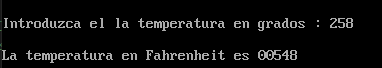
## [Inciso 2](#_INCISO_2)



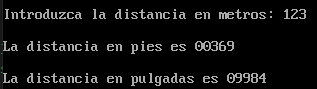
## [Inciso 4](#_INCISO_4)



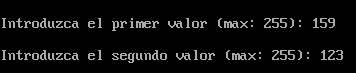
## [Inciso 5](#_INCISO_5)



## [Inciso 6](#_INCISO_6)



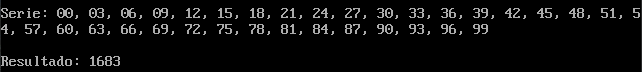
## [Inciso 7](#_INCISO_7)



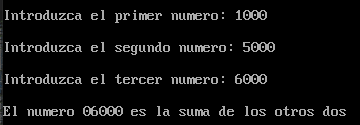
## [Inciso 8](#_INCISO_8)



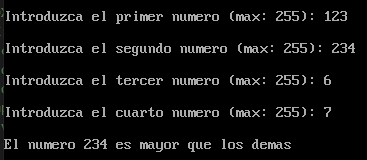
## [Inciso 9](#_INCISO_9)



## [Inciso 10](#_INCISO_10)



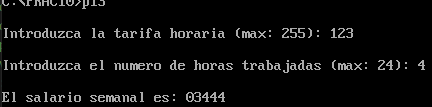
## [Inciso 11](#_INCISO_11)



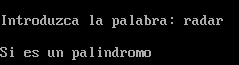
## [Inciso 12](#_INCISO_12)



## [Inciso 13](#_INCISO_13)



## [Inciso 14](#_INCISO_14)



## [Inciso 15](#_INCISO_15)



## [Inciso 16](#_INCISO_16)



## [Inciso 17](#_INCISO_17)

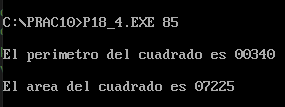


## Inciso 18

### [Inciso 2](#_18_2)



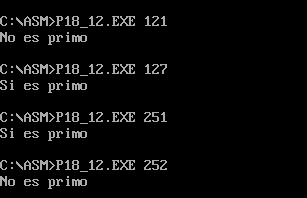
### [Inciso 4](#_18_4)



### [Inciso 10](#_18_10)



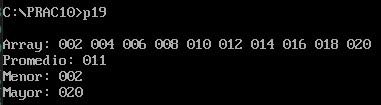
### [Inciso 12](#_18_12)



### [Inciso 14](#_18_14)



## [Inciso 19](#_INCISO_19)



## [Inciso 22](#_INCISO_22)



## [Inciso 23](#_INCISO_23)



# CONCLUSIONES

En esta práctica aprendimos a manejar a ensamblador a un nivel más avanzado, aprendimos a manejar interrupciones, la pila y limitar el uso de registros. Es muy interesante ver como a pesar de tener tantos registros, se llegan a dar casos en los que nos quedamos sin estos, y tenemos que intentar hacer más eficiente su uso.

# ANEXOS

## INCISO 1

MODEL small

.STACK 100h

LOCALS

.DATA

    message   db 10,13,'El cuadrado de 243 es: ',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov di,offset message

            call putStr

            mov al,0F3h

            mul al

            call printDec

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

            ENDP

; --- procedimientos ----;

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,5            ; inicializar conteo a 5 (decenas de mil-unidades de mil-cent-dec-unidades)

            mov bx,10000        ; iniciar con dec de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

        END

## INCISO 2

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Introduzca el valor a calcular (max: 255): ',0

    message   db 10,13,'El cuadrado es ',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax    ; del segmento de datos (.DATA)

            mov di,offset ask

            call putStr

            call getIntAL

            mov di,offset message

            call putStr

            mul al

            call printDec

            call putNewline

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,5            ; inicializar conteo a 5 (decenas de mil-unidades de mil-cent-dec-unidades)

            mov bx,10000        ; iniciar con dec de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getIntAL    PROC           ;Obtiene un valor y lo convierte a hexadecimal

            push bx

            push dx

            mov bx,0h

            mov dx,0Ah

@@while:    call getChar

            call isDec

            cmp bl,01  ;Mientras el digito introducido sea un decimal valido

            jne @@end

            sub al,'0'

            mov bl,al

            mov al,bh

            mul dl

            add al,bl

            mov bh,al

            jmp @@while

@@end:      mov al,bh

            pop dx

            pop bx

getIntAL    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC            ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'      ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h       ;En caso de haber un error, resetea bl

@@endCheck: ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC        ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline PROC             ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah     ;Salto de linea

            call putChar

            mov al, 0Dh     ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC            ;Funcion para imprimir un string

            push ax         ;Guarda registros a modificar

            push di

@@putStr:   mov al,[di]     ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h       ;Si es caracter de terminacion

            je @@endputStr  ;  Dejar de imprimir

            mov ah,0Eh      ;Selecciona el servicio 0Eh

            int 10h         ;Llama la interrupcion 10h

            inc di          ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC        ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 4

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Introduzca el lado a del cuadrado (max: 255): ',0

    messageP   db 10,13,'El perimetro del cuadrado es ',0

    messageA   db 10,13,'El area del cuadrado es ',0

    side dw 0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov di,offset ask

            call putStr

            call getInt

            mov bx,offset side

            mov [bx],ax

            mov di,offset messageP

            call putStr

            mov cl,4

            mul cl

            call printDec

            call putNewline

            mov di,offset messageA

            call putStr

            mov bx,offset side

            mov ax,[bx]

            mul al

            call printDec

            call putNewline

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,5            ; inicializar conteo a 5 (decenas de mil-unidades de mil-cent-dec-unidades)

            mov bx,10000        ; iniciar con dec de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getInt      PROC           ;Obtiene un valor y lo convierte a hexadecimal

            push bx

            push dx

            mov bx,0h

            mov dx,0Ah

@@while:    call getChar

            call isDec

            cmp bl,01  ;Mientras el digito introducido sea un decimal valido

            jne @@end

            sub al,'0'

            mov bl,al

            mov al,bh

            mul dl

            add al,bl

            mov bh,al

            jmp @@while

@@end:      mov al,bh

            pop dx

            pop bx

getInt      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC  ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck: ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar    ENDP

        END

## INCISO 5

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Introduzca el la temperatura en grados : ',0

    answer   db 10,13,'La temperatura en Fahrenheit es ',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov di,offset ask

            call putStr

            call getInt

            ;Formula F=(9/5)C+32

            ;9/5 = 1.8 redondeado a 2

            mov bx,2

            mul bx      ;2\*C

            add ax,32d  ;2\*C+32

            mov di,offset answer

            call putStr

            call printDec

            call putNewline

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

            ENDP

; --- procedimientos ----;

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,5            ; inicializar conteo a 5 (decenas de mil-unidades de mil-cent-dec-unidades)

            mov bx,10000        ; iniciar con dec de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            pop dx

            pop cx

            pop bx

            pop ax

            ret

            ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getInt   PROC           ;Obtiene un valor y lo convierte a hexadecimal

         push bx

         push cx

         push dx

         push si

           mov ax,0

           mov bx,0

         mov cx,10d

         mov dx,0

         mov si,0

@@next:  call getChar

        sub al,'0'

        cmp al,9

        ja @@end

         mov ah,0

         push ax     ;Guardamos el valor de ax en la pila

         mov ax,si

         mul cx

         pop dx      ;recuperamos el valor de ax y lo ponemos en dx

         add ax,dx

         mov si,ax

         inc bx

           jmp @@next

@@end:   mov ax,si

         pop si

         pop dx

         pop cx

         pop bx

         ret

         ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar  PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

         ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

         ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr   PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

         ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar  PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

         ENDP

        END

       ENDP

## INCISO 6

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Introduzca la distancia en metros: ',0

    answerF db 10,13,'La distancia en pies es ',0

    answerI db 10,13,'La distancia en pulgadas es ',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov di,offset ask

            call putStr

            call getInt

            mov bh,al

            ;1 Metro = 3 pies

            mov di,offset answerF

            call putStr

            mov bx,3

            mul bx

            call printDec

            call putNewline

            ;1 Metro = 39 pulgadas

            mov di,offset answerI

            call putStr

            mov al,bh

            mov bx,39d

            mul bx

            call printDec

            call putNewline

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,5            ; inicializar conteo a 5 (decenas de mil-unidades de mil-cent-dec-unidades)

            mov bx,10000        ; iniciar con dec de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getInt      PROC           ;Obtiene un valor y lo convierte a hexadecimal

            push bx

            push cx

            push dx

            push si

            mov ax,0

            mov bx,0

            mov cx,10d

            mov dx,0

            mov si,0

@@next:     call getChar

            sub al,'0'

            cmp al,9

            ja @@end

            mov ah,0

            push ax     ;Guardamos el valor de ax en la pila

            mov ax,si

            mul cx

            pop dx      ;recuperamos el valor de ax y lo ponemos en dx

            add ax,dx

            mov si,ax

            inc bx

            jmp @@next

@@end:      mov ax,si

            pop si

            pop dx

            pop cx

            pop bx

            ret

getInt      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 7

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Introduzca el primer valor (max: 255): ',0

    ask2 db 10,13,'Introduzca el segundo valor (max: 255): ',0

    answer db 10,13,'El mcd es ',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov di,offset ask

            call putStr

            call getInt

            mov bh,al

            mov di,offset ask2

            call putStr

            call getInt

            cmp al,0

            je @@end

            mov bl,al

            mov al,bh

            cmp al,0

            je @@end

            cmp bh,bl

            je @@end

            ja @@while

            mov al,bl

            mov bl,bh

@@while:    mov ah,0

            div bl

            cmp ah,0

            je @@end

            mov al,bl

            mov bl,ah

            jmp @@while

@@end:      mov al,bl

            mov di,offset answer

            call putStr

            call printDec

            call putNewline

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,3            ; inicializar conteo a 3 (cent-dec-unidades)

            mov ah,0

            mov bx,100        ; iniciar con cent

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getInt      PROC           ;Obtiene un valor y lo convierte a hexadecimal

            push bx

            push dx

            mov bx,0h

            mov dx,0Ah

@@while:    call getChar

            call isDec

            cmp bl,01  ;Mientras el digito introducido sea un decimal valido

            jne @@end

            sub al,'0'

            mov bl,al

            mov al,bh

            mul dl

            add al,bl

            mov bh,al

            jmp @@while

@@end:      mov al,bh

            pop dx

            pop bx

getInt      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC  ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck:

            ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 8

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Introduzca la serie de numeros: ',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov di, offset ask

            call putStr

@@while:    call getCh

            call isDec9

            cmp bl,01

            jl @@while

            jg @@end

            call putChar

            mov al,','

            call putChar

            mov al,' '

            call putChar

            jmp @@while

@@end:      call putNewline

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec9      PROC  ;Revisa que al sea un digito decimal valido (0-9)

            mov bl,1h

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jg @@endCheck

            mov bl,02h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck: ret

isDec9      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getCh       PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,8h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getCh       ENDP

        END

## INCISO 9

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Serie: ',0

    ans db 10,13,10,13,'Resultado: ',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov ax,0h

            mov bx,0h

            mov di, offset ask

            call putStr

            mov bx,10

            call printDecAL

            mov cl,1

            mov al,0

            mov bx,0

@@while:    mov al,','

            call putChar

            mov al,' '

            call putChar

            mov ax,3

            mul cl

            add bx,ax

            inc cl

            call printDecAL

            cmp cl,33d

            jle @@while

            mov di, offset ans

            call putStr

            mov ax,bx

            call printDecAX

            call putNewline

            mov ah,04ch         ; fin de programa

            mov al,0             ;

            int 21h              ;

@@end:      call putNewline

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

printDecAL  PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,2            ; inicializar conteo a 2 (cent-dec-unidades)

            mov bx,10          ; iniciar con centenas

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDecAL  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

printDecAX  PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,4            ; inicializar conteo a 4 (unidades de mil-cent-dec-unidades)

            mov bx,1000         ; iniciar con unid de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDecAX  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getCh       PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,8h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getCh       ENDP

        END

## INCISO 10

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Introduzca el primer numero: ',0

    ask2 db 10,13,'Introduzca el segundo numero: ',0

    ask3 db 10,13,'Introduzca el tercer numero: ',0

    num1  db 4 dup(0)

    num2  db 4 dup(0)

    num3  db 4 dup(0)

    ans db 10,13,'El numero ',0

    ans2 db 'es la suma de los otros dos',10,13,0

    ans3 db 10,13,'Ningun numero es la suma de los otros dos',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            lea di,ask

            call putStr

            call getInt

            lea bx,num1

            mov [bx],ax

            lea di,ask2

            call putStr

            call getInt

            lea si,num2

            mov [si],ax

            lea di,ask3

            call putStr

            call getInt

            mov [di],ax

            mov ax,[bx]

            mov ax,[si]

            mov ax,[di]

            mov ax,[bx]

            add ax,[si]

            cmp ax,[di]

            je @@case1

            mov ax,[di]

            add ax,[bx]

            cmp ax,[si]

            je @@case2

            mov ax,[si]

            add ax,[di]

            cmp ax,[bx]

            je @@case3

            jmp @@default

@@case1:    mov ax,[di]

            jmp @@print

@@case2:    mov ax,[si]

            jmp @@print

@@case3:    mov ax,[bx]

            jmp @@print

@@default   :  mov di,offset ans3

            call putStr

            jmp @@end

@@print:    mov di,offset ans

            call putStr

            call printDec

            mov di,offset ans2

            call putStr

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,5            ; inicializar conteo a 3 (cent-dec-unidades)

            mov bx,10000        ; iniciar con centenas

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            mov al,' '

            call putChar

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getInt      PROC

            push bx

            push cx

            push dx

            push si

            mov ax,0

            mov bx,0

            mov cx,10d

            mov dx,0

            mov si,0

@@next:     call getChar

            sub al,'0'

            cmp al,9

            ja @@end

            mov ah,0

            push ax     ;Guardamos el valor de ax en la pila

            mov ax,si

            mul cx

            pop dx      ;recuperamos el valor de ax y lo ponemos en dx

            add ax,dx

            mov si,ax

            inc bx

            jmp @@next

@@end:      mov ax,si

            pop si

            pop dx

            pop cx

            pop bx

            ret

getInt      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC  ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck: ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 11

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Introduzca el primer numero (max: 255): ',0

    ask2 db 10,13,'Introduzca el segundo numero (max: 255): ',0

    ask3 db 10,13,'Introduzca el tercer numero (max: 255): ',0

    ask4 db 10,13,'Introduzca el cuarto numero (max: 255): ',0

    ans db 10,13,'El numero ',0

    ans2 db 'es mayor que los demas',10,13,0

    greater db 0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov si,offset greater

            mov di,offset ask

            call putStr

            call getInt

            push ax

            mov di,offset ask2

            call putStr

            call getInt

            push ax

            mov di,offset ask3

            call putStr

            call getInt

            push ax

            mov di,offset ask4

            call putStr

            call getInt

            push ax

            mov cx,4

            pop ax

            mov [si],ax

@@while:    pop ax

            cmp ax,[si]

            ja @@exchange

            loop @@while

            jmp @@print

@@exchange: xchg ax,[si]

            loop @@while

@@print:    mov di,offset ans

            call putStr

            call printDec

            mov di,offset ans2

            call putStr

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

printDec    PROC

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,3            ; inicializar conteo a 3 (cent-dec-unidades)

            mov bx,100        ; iniciar con centenas

            mov ah,0

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            mov al,' '

            call putChar

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getInt      PROC           ;Obtiene un valor y lo convierte a hexadecimal

            push bx

            push dx

            mov bx,0h

            mov dx,0Ah

@@while:    call getChar

            call isDec

            cmp bl,01  ;Mientras el digito introducido sea un decimal valido

            jne @@end

            sub al,'0'

            mov bl,al

            mov al,bh

            mul dl

            add al,bl

            mov bh,al

            jmp @@while

@@end:      mov al,bh

            pop dx

            pop bx

getInt      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC  ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck:

            ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 12

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Introduzca el primer numero (max: 255): ',0

    ansSi db 'Si es primo',10,13,0

    ansNo db 'No es primo',10,13,0

    greater db 0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov si,offset greater

            mov di,offset ask

            call putStr

            call getInt

            mov ah,0

            mov [si], ax

            cmp al,0

            je @@case2

            cmp al,1

            je @@case1

            mov cx,2

            div cl

            inc cl

            cmp ah,0

            je @@case2

@@while:

            mov ax,[si]

            cmp cl,[si]

            jae @@case1

            div cl

            add cl,2

            cmp ah,0

            je @@case2

            jmp @@while

@@case1:    lea di,ansSi

            jmp @@print

@@case2:    lea di,ansNo

@@print:    call putStr

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

printDec    PROC

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,3            ; inicializar conteo a 3 (cent-dec-unidades)

            mov bx,100        ; iniciar con centenas

            mov ah,0

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            mov al,' '

            call putChar

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getInt      PROC           ;Obtiene un valor y lo convierte a hexadecimal

            push bx

            push dx

            mov bx,0h

            mov dx,0Ah

@@while:    call getChar

            call isDec

            cmp bl,01  ;Mientras el digito introducido sea un decimal valido

            jne @@end

            sub al,'0'

            mov bl,al

            mov al,bh

            mul dl

            add al,bl

            mov bh,al

            jmp @@while

@@end:      mov al,bh

            pop dx

            pop bx

getInt      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC  ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck: ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 13

MODEL small

.STACK 100h

LOCALS

.DATA

    askPay db 10,13,'Introduzca la tarifa horaria (max: 255): ',0

    askWork db 10,13,'Introduzca el numero de horas trabajadas (max: 24): ',0

    xcptn db 10,13,'El numero de horas no puede exceder las 24',10,13,0

    pay db 10,13,'El salario semanal es: ',0

    pay4work db 0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov cx,07

            lea di, askPay

            call putStr

            call getInt

            lea si, pay4work

            mov [si],al

            lea di, askWork

            call putStr

            call getInt

            cmp al,24

            ja @@exception

            mul cl

            mov cl,[si]

            mul cl

            lea di, pay

            jmp @@print

@@exception:lea di,xcptn

            call putStr

            jmp @@end

@@print:    call putStr

            call printDec

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

printDec    PROC

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,5            ; inicializar conteo a 3 (cent-dec-unidades)

            mov bx,10000        ; iniciar con centenas

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            mov al,' '

            call putChar

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getInt      PROC           ;Obtiene un valor y lo convierte a hexadecimal

            push bx

            push dx

            mov bx,0h

            mov dx,0Ah

@@while:    call getChar

            call isDec

            cmp bl,01  ;Mientras el digito introducido sea un decimal valido

            jne @@end

            sub al,'0'

            mov bl,al

            mov al,bh

            mul dl

            add al,bl

            mov bh,al

            jmp @@while

@@end:      mov al,bh

            pop dx

            pop bx

getInt      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC  ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck:

            ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 14

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Introduzca la palabra: ',0

    pal db 10,13,'No es un palindromo',10,13,0

    noPal db 10,13,'Si es un palindromo',10,13,0

    palWord db 21 dup(0)

    revPalWord db 21 dup(0)

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            lea di,ask

            call putStr

            lea di,palWord

            call getAlphaStr

            lea di,palWord

            lea si,revPalWord

@@prepareDI:

            cmp byte ptr [di],0

            je @@invertStr

            inc di

            jmp @@prepareDI

@@invertStr:

            dec di

            cmp di,offset palWord-1

            je @@endRev

            mov al,[di]

            mov [si],al

            inc si

            jmp @@invertStr

@@endStr:   mov byte ptr [si],0

@@endRev:   lea si,revPalWord

            lea di,palWord

            call palindrome

            cmp bl,1

            je @@notPal

            lea di,pal

            jmp @@print

@@notPal:   lea di,noPal

@@print:    call putStr

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

palindrome  PROC

            push ax

            push di

            push si

            mov bl,0    ;Empezamos suponiendo que no son palindromos

@@compareStr:

            mov al,[di]

            cmp al,0

            je @@isPal

            mov al,[si]

            cmp al,0

            je @@isPal

            cmp [di],al

            jne @@end

            inc si

            inc di

            jmp @@compareStr

@@isPal:    mov bl,1

@@end:      pop si

            pop di

            pop ax

            ret

palindrome  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getAlphaStr PROC

            push ax

            push cx

            push di

            mov cx,20

@@while:    call getChar

            call isAplha

            cmp bl,02

            je @@end2

            mov [di],al

            inc di

            dec cx

            cmp cx,0

            je @@end

            jmp @@while

@@end2:     inc di

@@end:      mov byte ptr [di],0

            pop di

            pop cx

            pop ax

            ret

getAlphaStr ENDP              ;El tamaño del string se encuentra en la segunda posicion

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isAplha     PROC  ;Si no esta dentro devuelve 02h en bl, si es mayuscula 01h, si es minuscula 00h

            push ax

            ;Para que este en el alfabeto mayuscula debe cumplirse ('A' >= x <= 'Z')

            cmp al,'A'

            jl @@exception

            cmp al,'Z'

            jl @@isUpper

            ;Para que este en el alfabeto minuscula debe cumplirse ('a' >= x <= 'z')

            cmp al,'a'

            jl @@exception

            cmp al,'z'

            jl @@isLower

            ;Si no se cumple ningun caso anterior, no esta en el alfabeto

            jmp @@exception

@@exception:mov bl,2h

            jmp @@endCheck

@@isUpper:  mov bl,1h

            jmp @@endCheck

@@isLower:  mov bl,0h

@@endCheck: pop ax   ;Recupera registro

            ret

isAplha     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 15

MODEL small

.STACK 100h

LOCALS

.DATA

    ask db 10,13,'Introduzca la palabra: ',0

    palabra db 27 dup(0)

    alpha db 27 dup(0)

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            lea di,ask

            call putStr

            lea di,palabra

            call getAlphaStr

            lea si,alpha

            call count

            call putNewline

            mov bx,0

@@while:    cmp byte ptr si[bx],0

            je @@continue

@@printCount:

            mov al,bl

            add al,'A'

            call putChar

            mov al,'='

            call putChar

            mov al,si[bx]

            call printDec

@@continue: inc bx

            cmp bx,26

            jb @@while

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

count       PROC

            push ax

            push bx

            push di

            push si

@@countLetters:

            mov al,[di]

            call isAplha

            cmp bl,2

            je @@end

            cmp bl,1

            je @@upper2Lower

            jmp @@count

@@upper2Lower:

            add al,' '

            mov [di],al

@@count:

            mov bh,0

            mov bl,[di]

            sub bl,'a'

            mov ah,0

            add byte ptr si[bx],01

            inc di

            jmp @@countLetters

@@end:      pop si

            pop di

            pop bx

            pop ax

            ret

count       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,3            ; inicializar conteo a 3 (cent-dec-unidades)

            mov bx,100        ; iniciar con centenas

            mov ah,0

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            mov al,' '

            call putChar

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getAlphaStr PROC

            push ax

            push cx

            push di

            mov cx,20

@@while:

            call getChar

            call isAplha

            cmp bl,02

            je @@end2

            mov [di],al

            inc di

            dec cx

            cmp cx,0

            je @@end

            jmp @@while

@@end2:     inc di

@@end:      mov byte ptr [di],0

            pop di

            pop cx

            pop ax

            ret

getAlphaStr ENDP              ;El tamaño del string se encuentra en la segunda posicion

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isAplha     PROC  ;Si no esta dentro devuelve 02h en bl, si es mayuscula 01h, si es minuscula 00h

            push ax

            ;Para que este en el alfabeto mayuscula debe cumplirse ('A' >= x <= 'Z')

            cmp al,'A'

            jl @@exception

            cmp al,'Z'

            jl @@isUpper

            ;Para que este en el alfabeto minuscula debe cumplirse ('a' >= x <= 'z')

            cmp al,'a'

            jl @@exception

            cmp al,'z'

            jl @@isLower

            ;Si no se cumple ningun caso anterior, no esta en el alfabeto

            jmp @@exception

@@exception:mov bl,2h

            jmp @@endCheck

@@isUpper:  mov bl,1h

            jmp @@endCheck

@@isLower:  mov bl,0h

@@endCheck: pop ax   ;Recupera registro

            ret

isAplha     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 16

MODEL small

.STACK 100h

LOCALS

.DATA

    number db 4 dup (0)

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov ax,1234d

            call printDec

            mov al,'='

            call putChar

            mov al,' '

            call putChar

            mov ax,1234d

            lea di,number

            call hex2str

            call putStr

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,4            ; inicializar conteo a 4 (mil-cent-dec-unidades)

            mov bx,1000        ; iniciar con millares

            mov ah,0

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            mov al,' '

            call putChar

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

hex2str     PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax            ; salvar registro a utilizar

            push bx

            push cx

            push dx

            push di

            push si

            mov bx,0

            mov cx,4            ; inicializar conteo a 4 (mil-cent-dec-unidades)

            mov si,1000         ; iniciar con millares

            mov ah,0

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div si              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            mov di[bx],al   ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,si           ; la idea es:

            mov si,10           ; BX = BX/10

            div si

            mov si,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            inc bx

            loop @@nxt          ; proximo digito

            mov byte ptr di[bx],0

            pop si

            pop di

            pop dx

            pop cx

            pop bx

            pop ax

            ret

hex2str     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getAlphaStr PROC

            push ax

            push cx

            push di

            mov cx,20

@@while:    call getChar

            call isAplha

            cmp bl,02

            je @@end2

            mov [di],al

            inc di

            dec cx

            cmp cx,0

            je @@end

            jmp @@while

@@end2:     inc di

@@end:      mov byte ptr [di],0

            pop di

            pop cx

            pop ax

            ret

getAlphaStr ENDP              ;El tamaño del string se encuentra en la segunda posicion

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isAplha     PROC  ;Si no esta dentro devuelve 02h en bl, si es mayuscula 01h, si es minuscula 00h

            push ax

            ;Para que este en el alfabeto mayuscula debe cumplirse ('A' >= x <= 'Z')

            cmp al,'A'

            jl @@exception

            cmp al,'Z'

            jl @@isUpper

            ;Para que este en el alfabeto minuscula debe cumplirse ('a' >= x <= 'z')

            cmp al,'a'

            jl @@exception

            cmp al,'z'

            jl @@isLower

            ;Si no se cumple ningun caso anterior, no esta en el alfabeto

            jmp @@exception

@@exception:mov bl,2h

            jmp @@endCheck

@@isUpper:  mov bl,1h

            jmp @@endCheck

@@isLower:  mov bl,0h

@@endCheck: pop ax   ;Recupera registro

            ret

isAplha     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 17

MODEL small

.STACK 100h

LOCALS

.DATA

        number db '1000',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            lea di, number

            call putStr

            mov al,'='

            lea di,number

            call putChar

            call text2dec

            call printDec

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

text2dec    PROC

            push bx

            push cx

            push dx

            push si

            mov ax,0

            mov bx,0

            mov cx,10d

            mov dx,0

            mov si,0

@@next:     mov al,byte ptr di[bx]

            sub al,'0'

            cmp al,9

            ja @@end

            mov ah,0

            push ax     ;Guardamos el valor de ax en la pila

            mov ax,si

            mul cx

            pop dx      ;recuperamos el valor de ax y lo ponemos en dx

            add ax,dx

            mov si,ax

            inc bx

            jmp @@next

@@end:      mov ax,si

            pop si

            pop dx

            pop cx

            pop bx

            ret

text2dec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,4            ; inicializar conteo a 4 (mil-cent-dec-unidades)

            mov bx,1000        ; iniciar con millares

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            mov al,' '

            call putChar

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getAlphaStr PROC

            push ax

            push cx

            push di

            mov cx,20

@@while:    call getChar

            call isAplha

            cmp bl,02

            je @@end2

            mov [di],al

            inc di

            dec cx

            cmp cx,0

            je @@end

            jmp @@while

@@end2:     inc di

@@end:      mov byte ptr [di],0

            pop di

            pop cx

            pop ax

            ret

getAlphaStr ENDP              ;El tamaño del string se encuentra en la segunda posicion

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isAplha     PROC  ;Si no esta dentro devuelve 02h en bl, si es mayuscula 01h, si es minuscula 00h

            push ax

            ;Para que este en el alfabeto mayuscula debe cumplirse ('A' >= x <= 'Z')

            cmp al,'A'

            jl @@exception

            cmp al,'Z'

            jl @@isUpper

            ;Para que este en el alfabeto minuscula debe cumplirse ('a' >= x <= 'z')

            cmp al,'a'

            jl @@exception

            cmp al,'z'

            jl @@isLower

            ;Si no se cumple ningun caso anterior, no esta en el alfabeto

            jmp @@exception

@@exception:mov bl,2h

            jmp @@endCheck

@@isUpper:  mov bl,1h

            jmp @@endCheck

@@isLower:  mov bl,0h

@@endCheck: pop ax   ;Recupera registro

            ret

isAplha     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 18

### 18\_2

MODEL small

.STACK 100h

LOCALS

    ARGSSIZE EQU 20

.DATA

    message   db 10,13,'El cuadrado es ',0

    args  db ARGSSIZE dup(0)

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov cl, ds:[80]

            mov dx, 81h

            call cleanPar

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            call changeSegment

            lea di,message

            call putStr

            lea di,args

            call text2dec

            mul al

            call printDec

            call putNewline

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

Principal   ENDP

; --- procedimientos ----;

changeSegment PROC   ;Los argumentos por linea de comandos se encuentran

                     ;en el segmento extra, para facilitar la manipulacion

                     ;se mueven al segmento de datos

            mov cx,ARGSSIZE

            mov di,0

@@while:

            mov al,es:args[di]   ;Se copia el valor de es:args[di] a al

            mov args[di],al      ;Se copia el valor de al a ds:args[di]

            loop @@while

            ret

changeSegment ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cleanPar    PROC

            push di

            push bx

            mov ax,@data

            mov es,ax

            mov di,0

            mov bx,dx

            mov al,[bx + di + 1] ; se suma 1 porque el primero es un espacio

@@loop:     mov es:args[di], al

            inc di

            mov al,[bx + di + 1]

            ; call putchar

            cmp al,0

            loopne @@loop

            pop bx

            pop di

            ret

cleanPar    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

text2dec    PROC

            push bx

            push cx

            push dx

            push si

            mov ax,0

            mov bx,0

            mov cx,10d

            mov dx,0

            mov si,0

@@next:     mov al,byte ptr di[bx]

            sub al,'0'

            cmp al,9

            ja @@end

            mov ah,0

            push ax     ;Guardamos el valor de ax en la pila

            mov ax,si

            mul cx

            pop dx      ;recuperamos el valor de ax y lo ponemos en dx

            add ax,dx

            mov si,ax

            inc bx

            jmp @@next

@@end:      mov ax,si

            pop si

            pop dx

            pop cx

            pop bx

            ret

text2dec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,5            ; inicializar conteo a 5 (decenas de mil-unidades de mil-cent-dec-unidades)

            mov bx,10000        ; iniciar con dec de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC  ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck: ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

### 18\_4

MODEL small

.STACK 100h

LOCALS

    ARGSSIZE EQU 20

.DATA

    messageP   db 10,13,'El perimetro del cuadrado es ',0

    messageA   db 10,13,'El area del cuadrado es ',0

    args  db ARGSSIZE dup(0)

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov cl, ds:[80]

            mov dx, 81h

            call cleanPar

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            call changeSegment

            lea di,args

            call text2dec

            mov bx,offset args

            mov [bx],ax

            mov di,offset messageP

            call putStr

            mov cx,4

            mul cx

            call printDec

            call putNewline

            mov di,offset messageA

            call putStr

            mov bx,offset args

            mov ax,[bx]

            mul ax

            call printDec

            call putNewline

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

Principal   ENDP

; --- procedimientos ----;

getArg      PROC

            push ax

            push di

            mov cx,ARGSSIZE

            mov di,0

@@loop:     mov al,args[di]

            cmp al,' '

            jne @@copy

            jmp @@end

@@copy:     mov args[di],al

            inc di

            loop @@loop

@@end:      pop di

            pop ax

            ret

getArg      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changeSegment PROC   ;Los argumentos por linea de comandos se encuentran

                     ;en el segmento extra, para facilitar la manipulacion

                     ;se mueven al segmento de datos

            mov cx,ARGSSIZE

            mov di,0

@@while:

            mov al,es:args[di]   ;Se copia el valor de es:args[di] a al

            mov args[di],al      ;Se copia el valor de al a ds:args[di]

            loop @@while

            ret

changeSegment ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cleanPar    PROC

            push di

            push bx

            mov ax,@data

            mov es,ax

            mov di,0

            mov bx,dx

            mov al,[bx + di + 1] ; se suma 1 porque el primero es un espacio

@@loop:     mov es:args[di], al

            inc di

            mov al,[bx + di + 1]

            cmp al,0

            loopne @@loop

            pop bx

            pop di

            ret

cleanPar    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

text2dec    PROC

            push bx

            push cx

            push dx

            push si

            mov ax,0

            mov bx,0

            mov cx,10d

            mov dx,0

            mov si,0

@@next:     mov al,byte ptr di[bx]

            sub al,'0'

            cmp al,9

            ja @@end

            mov ah,0

            push ax     ;Guardamos el valor de ax en la pila

            mov ax,si

            mul cx

            pop dx      ;recuperamos el valor de ax y lo ponemos en dx

            add ax,dx

            mov si,ax

            inc bx

            jmp @@next

@@end:      mov ax,si

            pop si

            pop dx

            pop cx

            pop bx

            ret

text2dec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,5            ; inicializar conteo a 5 (decenas de mil-unidades de mil-cent-dec-unidades)

            mov bx,10000        ; iniciar con dec de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC  ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck: ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

### 18\_10

MODEL small

.STACK 100h

LOCALS

    ARGSSIZE EQU 20

.DATA

    args  db ARGSSIZE dup(0)

    arg1  db ARGSSIZE dup(0)

    arg2  db ARGSSIZE dup(0)

    ans db 10,13,'El numero ',0

    ans2 db 'es la suma de los otros dos',10,13,0

    ans3 db 10,13,'Ningun numero es la suma de los otros dos',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov cl, ds:[80]     ;

            mov dx, 81h         ; en esta direccion

            call cleanPar       ; elimina el espacio y pasa los argumentos a una var

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            call changeSegment

            call separateArgs

            lea di,args

            call text2dec

            lea bx,args

            mov [bx],ax

            call printDec

            lea di,arg1

            call text2dec

            lea si,arg1

            mov [si],ax

            call printDec

            lea di,arg2

            call text2dec

            mov [di],ax

            call printDec

            mov ax,[bx]

            mov ax,[si]

            mov ax,[di]

            mov ax,[bx]

            add ax,[si]

            cmp ax,[di]

            je @@case1

            mov ax,[di]

            add ax,[bx]

            cmp ax,[si]

            je @@case2

            mov ax,[si]

            add ax,[di]

            cmp ax,[bx]

            je @@case3

            jmp @@default

@@case1:    mov ax,[di]

            jmp @@print

@@case2:    mov ax,[si]

            jmp @@print

@@case3:    mov ax,[bx]

            jmp @@print

@@default:  mov di,offset ans3

            call putStr

            jmp @@end

@@print:    mov di,offset ans

            call putStr

            call printDec

            mov di,offset ans2

            call putStr

@@end:      call putNewline

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

Principal   ENDP

; --- procedimientos ----;

separateArgs PROC

            push ax

            push bx

            push cx

            push dx

            push di

            push si

            mov cx,ARGSSIZE

            mov bx,0

            mov si,0

            mov dx,0

@@while:

            mov al,args[di]

            cmp al,' '

            je @@incDX

            cmp dx,0

            je @@incDI

            cmp al,0

            je @@end

            cmp dx,1

            je @@cpy2arg2

            mov arg2[si],al

            inc si

            jmp @@incDI

@@cpy2arg2: mov arg1[bx],al

            inc bx

            jmp @@incDI

@@incDX:    inc dx

@@incDI:    inc di

            loop @@while

@@end:      pop si

            pop di

            pop dx

            pop cx

            pop bx

            pop ax

            ret

separateArgs ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changeSegment PROC   ;Los argumentos por linea de comandos se encuentran

                     ;en el segmento extra, para facilitar la manipulacion

                     ;se mueven al segmento de datos

            mov cx,ARGSSIZE

            mov di,0

@@while:    mov al,es:args[di]   ;Se copia el valor de es:args[di] a al

            mov args[di],al      ;Se copia el valor de al a ds:args[di]

            loop @@while

            ret

changeSegment ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cleanPar    PROC

            push di

            push bx

            mov ax,@data

            mov es,ax

            mov di,0

            mov bx,dx

            mov al,[bx + di + 1] ; se suma 1 porque el primero es un espacio

    @@loop: mov es:args[di], al

            inc di

            mov al,[bx + di + 1]

            cmp al,0

            loopne @@loop

            pop bx

            pop di

            ret

cleanPar    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

text2dec    PROC

            push bx

            push cx

            push dx

            push si

            mov ax,0

            mov bx,0

            mov cx,10d

            mov dx,0

            mov si,0

@@next:     mov al,byte ptr di[bx]

            sub al,'0'

            cmp al,9

            ja @@end

            mov ah,0

            push ax     ;Guardamos el valor de ax en la pila

            mov ax,si

            mul cx

            pop dx      ;recuperamos el valor de ax y lo ponemos en dx

            add ax,dx

            mov si,ax

            inc bx

            jmp @@next

@@end:      mov ax,si

            pop si

            pop dx

            pop cx

            pop bx

            ret

text2dec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,5            ; inicializar conteo a 5 (decenas de mil-unidades de mil-cent-dec-unidades)

            mov bx,10000        ; iniciar con dec de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            mov ax,' '

            call putchar

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC  ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck: ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

### 18\_12

MODEL small

.STACK 100h

LOCALS

    ARGSSIZE EQU 20

.DATA

    ansSi db 'Si es primo',10,13,0

    ansNo db 'No es primo',10,13,0

    args  db ARGSSIZE dup(0)

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov cl, ds:[80]     ;

            mov dx, 81h         ; en esta direccion

            call cleanPar       ; elimina el espacio y pasa los argumentos a una var

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            call changeSegment

            lea di,args

            call text2dec

            mov [di], ax

            cmp al,0

            je @@case2

            cmp al,1

            je @@case1

            mov cx,2

            div cl

            inc cl

            cmp ah,0

            je @@case2

@@while:

            mov ax,[di]

            cmp cx,[di]

            jae @@case1

            div cl

            add cl,2

            cmp ah,0

            je @@case2

            jmp @@while

@@case1:    lea di,ansSi

            jmp @@print

@@case2:    lea di,ansNo

@@print:    call putStr

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

Principal   ENDP

; --- procedimientos ----;

changeSegment PROC   ;Los argumentos por linea de comandos se encuentran

                     ;en el segmento extra, para facilitar la manipulacion

                     ;se mueven al segmento de datos

            mov cx,ARGSSIZE

            mov di,0

@@while:

            mov al,es:args[di]   ;Se copia el valor de es:args[di] a al

            mov args[di],al      ;Se copia el valor de al a ds:args[di]

            loop @@while

            ret

changeSegment ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cleanPar    PROC

            push di

            push bx

            mov ax,@data

            mov es,ax

            mov di,0

            mov bx,dx

            mov al,[bx + di + 1] ; se suma 1 porque el primero es un espacio

    @@loop:

            mov es:args[di], al

            inc di

            mov al,[bx + di + 1]

            ; call putchar

            cmp al,0

            loopne @@loop

            pop bx

            pop di

            ret

cleanPar    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

text2dec    PROC

            push bx

            push cx

            push dx

            push si

            mov ax,0

            mov bx,0

            mov cx,10d

            mov dx,0

            mov si,0

@@next:     mov al,byte ptr di[bx]

            sub al,'0'

            cmp al,9

            ja @@end

            mov ah,0

            push ax     ;Guardamos el valor de ax en la pila

            mov ax,si

            mul cx

            pop dx      ;recuperamos el valor de ax y lo ponemos en dx

            add ax,dx

            mov si,ax

            inc bx

            jmp @@next

@@end:      mov ax,si

            pop si

            pop dx

            pop cx

            pop bx

            ret

text2dec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,5            ; inicializar conteo a 5 (decenas de mil-unidades de mil-cent-dec-unidades)

            mov bx,10000        ; iniciar con dec de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            mov ax,' '

            call putchar

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC  ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck: ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

### 18\_14

MODEL small

.STACK 100h

LOCALS

    ARGSSIZE EQU 20

.DATA

         args  db ARGSSIZE dup(0)

         pal db 10,13,'No es un palindromo',10,13,0

         noPal db 10,13,'Si es un palindromo',10,13,0

         revPalWord db ARGSSIZE dup(0)

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov cl, ds:[80]

            mov dx, 81h

            call cleanPar

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            lea di,args

            call changeSegment

            lea si,revPalWord

@@prepareDI:

            cmp byte ptr [di],0

            je @@invertStr

            inc di

            jmp @@prepareDI

@@invertStr:

            dec di

            cmp di,offset args-1

            je @@endRev

            mov al,[di]

            mov [si],al

            inc si

            jmp @@invertStr

@@endStr:   mov byte ptr [si],0

@@endRev:   lea si,revPalWord

            lea di,args

            call palindrome

            cmp bl,1

            je @@notPal

            lea di,pal

            jmp @@print

@@notPal:   lea di,noPal

@@print:    call putStr

@@end:      mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

changeSegment PROC   ;Los argumentos por linea de comandos se encuentran

                     ;en el segmento extra, para facilitar la manipulacion

                     ;se mueven al segmento de datos

            mov cx,ARGSSIZE

            mov bx,0

@@while:    mov al,es:args[bx]   ;Se copia el valor de es:args[di] a al

            mov args[bx],al      ;Se copia el valor de al a ds:args[di]

            loop @@while

            ret

changeSegment ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cleanPar    PROC

            push di

            push bx

            mov ax,@data

            mov es,ax

            mov di,0

            mov bx,dx

            mov al,[bx + di + 1] ; se suma 1 porque el primero es un espacio

@@loop:

            mov es:args[di], al

            inc di

            mov al,[bx + di + 1]

            cmp al,0

            loopne @@loop

            pop bx

            pop di

            ret

cleanPar    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getArg      PROC

            push ax

            push si

            mov cx,ARGSSIZE

            mov si,1

@@loop:

            mov al,args[si]

            cmp al,' '

            jne @@copy

            jmp @@end

@@copy:

            mov di[bx],al

            inc si

            inc bx

            loop @@loop

@@end:

            pop di

            pop ax

            ret

getArg      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

palindrome  PROC

            push ax

            push cx

            push si

            mov si,0

            mov cx,ARGSSIZE

            mov bx,0    ;Empezamos suponiendo que no son palindromos

@@compareStr:

            mov al,revPalWord[si+1]

            cmp al,0

            je @@isPal

            cmp args[si],al

            jne @@end

            inc si

            loop @@compareStr

@@isPal:    mov bl,1

@@end:      pop si

            pop cx

            pop ax

            ret

palindrome  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,3            ; inicializar conteo a 5 (decenas de mil-unidades de mil-cent-dec-unidades)

            mov bx,100        ; iniciar con dec de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            mov ax,' '

            call putchar

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

isDec       PROC  ;Revisa que al sea un digito decimal valido (0-9)

            cmp al,'9'            ;Compara que al se encuentre dentro del rango (0-9)

            jg @@exception

            cmp al,'0'

            jl @@exception

            mov bl,1h

            jmp @@endCheck

@@exception:

            mov bl,0h           ;En caso de haber un error, resetea bl

@@endCheck:

            ret

isDec       ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

getChar     PROC     ;Funcion para leer un cahar y almacenarlo en al

            mov ah,1h   ;Selecciona el servicio 01h

            int 21h     ;Llama la interrupcion 21h

            ret

getChar     ENDP

        END

## INCISO 19

MODEL small

.STACK 100h

LOCALS

.DATA

    array db 2,4,6,8,10,12,14,16,18,20

    array\_message db 10,13,'Array: ',0

    avg db 10,13,'Promedio: ',0

    maximum db 10,13,'Mayor: ',0

    minimum db 10,13,'Menor: ',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov cx,10

            lea di,array\_message

            call putStr

            call printArray

            lea di,avg

            call putStr

            call average

            call printDec

            lea di,minimum

            call putStr

            call min

            call printDec

            lea di,maximum

            call putStr

            call max

            call printDec

            call putNewline

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

printArray  PROC

            push bx

            push cx

            mov bx,0

            mov ah,0

@@print:    mov al,array[bx]

            call printDec

            mov al,' '

            call putChar

            inc bx

            loop @@print

            pop cx

            pop bx

            ret

printArray  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

average     PROC

            push bx

            push dx

            push cx

            mov ax,0

            mov bx,0

@@add:      add al,array[bx]

            inc bx

            loop @@add

            pop cx

            div cl

            pop dx

            pop bx

            ret

average     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

max         PROC

            push bx

            push dx

            push cx

            mov ax,0

            mov bx,0

@@add:      cmp al,array[bx]

            jae @@incBX

            mov al,array[bx]

@@incBX:    inc bx

            loop @@add

            pop cx

            pop dx

            pop bx

            ret

max         ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

min         PROC

            push bx

            push dx

            push cx

            mov bx,0

            mov ax,0

            mov al,array[bx]

            inc bx

            dec cx

@@add:      cmp al,array[bx]

            jbe @@incBX

            mov al,array[bx]

@@incBX:    inc bx

            loop @@add

            pop cx

            pop dx

            pop bx

            ret

min         ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

printDec    PROC                ;Modificado para imprimir todo AX en vez de solo AL

            push ax             ; salvar registro a utilizar

            push bx

            push cx

            push dx

            mov cx,3            ; inicializar conteo a 5 (decenas de mil-unidades de mil-cent-dec-unidades)

            mov bx,100        ; iniciar con dec de millar

@@nxt:      mov dx,0            ; asegurar DX=0 para usar div reg16

            div bx              ; dividir DX:AX entre BX

            add al,'0'          ; convertir cociente a ASCII

            call putChar        ; desplegar digito en pantalla

            mov ax,dx           ; pasar residuo (DX) a AX

            push ax             ; salvar temporalmente AX

            mov dx,0            ; ajustar divisor para nuevo digito

            mov ax,bx           ; la idea es:

            mov bx,10           ; BX = BX/10

            div bx

            mov bx,ax           ; pasar cociente al BX para nuevo digito

            pop ax              ; recupera AX

            loop @@nxt          ; proximo digito

            pop dx

            pop cx

            pop bx

            pop ax

            ret

printDec    ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putChar     PROC     ;Funcion para imprimir un char almacenado en al

            push ax     ;Guarda el valor del registro a modificar

            mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;Llama la interrupcion 10h

            pop ax      ;Recupera registro

            ret

putChar     ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putNewline  PROC   ;Funcion  para imprimir un salto de linea

            push ax

            mov al, 0Ah    ;Salto de linea

            call putChar

            mov al, 0Dh    ;Retorno de carro

            call putChar

            pop ax

            ret

putNewline  ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putStr      PROC      ;Funcion para imprimir un string

            push ax        ;Guarda registros a modificar

            push di

@@putStr:

            mov al,[di]    ;Obtiene el valor de la direccion di y la guarda en al

            cmp al,0h      ;Si es caracter de terminacion

            je @@endputStr ;  Dejar de imprimir

            mov ah,0Eh     ;Selecciona el servicio 0Eh

            int 10h        ;Llama la interrupcion 10h

            inc di         ;decrementa di

            jmp @@putStr

@@endputStr:

            pop di         ;Recupera registros modificados

            pop ax

            ret

putStr      ENDP

        END

## INCISO 22

MODEL small

.STACK 100h

LOCALS

.DATA

    string db 10,13,'Cadena apuntada por BX',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov ax,0

            mov es,ax   ;Las interrupciones se encuentran desde 0 a 3FF;

            CLI         ;Desactiva la bandera de interrupciones

            mov word ptr es:[88h], offset int22     ;Se pone la direccion de la interrupcion en el vector

            mov word ptr es:[8Ah], cs               ;Indicamos en donde buscar el codigo (segmento de codigo)

            STI         ;Vuelve a activarlas

            mov ah,1    ;Selecciona la interrupcion 1

            mov al,'$'  ;Pone el char $ en al

            int 22h     ;Llama la interrupcion 23 (la creada)

            lea bx,string   ;Apunta bx al string

            mov ah,2        ;Selecciona la interrupcion 1

            int 22h         ;Llama la interrupcion 23

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int22       PROC FAR

            push ax

            push dx

            push bx

            cmp ah,1        ;Servicio 1

            je @@putchar

            cmp ah,2        ;Servicio 2

            je @@puts

            jmp @@end

@@putchar:  mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;llama la interrupción para imprimir char

            jmp @@end

@@puts:     mov ah,0Eh  ;Selecciona el servicio 0Eh

@@loop:     mov al,[bx] ;Obtiene el char

            cmp al,0    ;Si es 0, deja de imprimir

            je @@end

            int 10h     ;si no es 0, imprime

            inc bx      ;aumenta para siguiente char

            jmp @@loop

@@end:      pop bx

            pop dx

            pop ax

            iret

int22       ENDP

    END

## INCISO 23

MODEL small

.STACK 100h

        ;----- Insert INCLUDE "filename" directives here

        ;----- Insert EQU and = equates here

       LOCALS

.DATA

         string db 10,13,'Cadena apuntada por BX',0

.CODE ;----- Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data ;Inicializar DS al la direccion

            mov ds,ax ; del segmento de datos (.DATA)

            mov ax,8Ch      ;Valor de donde sera cargada la interrupcion

            lea bx,Int23    ;Desplazamiento de la RSI

            mov dx,cs       ;Segmento, la RSI se encuentra en el segmento de codigo

            call putInt

            mov ah,1    ;Selecciona la interrupcion 1

            mov al,'$'  ;Pone el char $ en al

            int 23h     ;Llama la interrupcion 23 (la creada)

            lea bx,string   ;Apunta bx al string

            mov ah,2        ;Selecciona la interrupcion 1

            int 23h         ;Llama la interrupcion 23

            mov ah,04ch ; fin de programa

            mov al,0

            int 21h

            ret

Principal   ENDP

; --- procedimientos ----;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

putInt      PROC

            push ax

            push di

            push es

            mov di, ax      ;Copiamos el valor de ax a un registro base o indice

            mov ax,0

            mov es,ax   ;Las interrupciones se encuentran desde 0 a 3FF;

            CLI         ;Desactiva interrupciones

            mov word ptr es:[di], offset int23      ;Se pone la direccion de la interrupcion en el vector

            mov word ptr es:[di+2], dx              ;Indicamos en donde buscar el codigo

            STI         ;Vuelve a activarlas

            pop es

            pop ax

            pop si

            ret

putInt      ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Int23       PROC FAR

            push ax

            push dx

            push bx

            cmp ah,1        ;Servicio 1

            je @@putchar

            cmp ah,2        ;Servicio 2

            je @@puts

            jmp @@end

@@putchar:  mov ah,0Eh  ;Selecciona el servicio 0Eh

            int 10h     ;llama la interrupción para imprimir char

            jmp @@end

@@puts:     mov ah,0Eh  ;Selecciona el servicio 0Eh

@@loop:     mov al,[bx] ;Obtiene el char

            cmp al,0    ;Si es 0, deja de imprimir

            je @@end

            int 10h     ;si no es 0, imprime

            inc bx      ;aumenta para siguiente char

            jmp @@loop

@@end:      pop bx

            pop dx

            pop ax

            iret

Int23       ENDP

        END