Listado de Programas
Autor: Nery Guzmán

No estan en orden de dificultad pero con las descripciones mas que todo servirá como referencia a algunas interrupciones clásicas en un curso de assembler.

1.

```
;EXCEPCION AH07 de la int 21 h
;programa que lee una tecla del buffer sin repeticion en pantalla( sin eco)
; luego se manda con la excepcion ah07 dentro de al y se compara
; siendo 00 tecla de funcion
```

2.

```
;EXCEPCION AH07
; programa que hace 2cuadros en el centro de la pantalla, posiciona el cursor y
deja escribir 20 caracteres sin verificar
; por  medio del manejador de teclado
.MODEL SMALL
.STACK 64
```

3.

```
;este programa lo que hace es agarrar un arreglo que ingresa el usuario y le
quita todos los espacios en blanco
; y lo imprime de nuevo sin los espacios ( aca se debio haber usado scas y stows
pero yo utilize escritura directa en memoria
; para explorar y reescribir los arreglos)
```

4.

```
;File Writing by 414C485649
;     Escribe un arcrivo en el disco utilizando las funciones que provee el DOS
con la int 21h
;Info y ejemplo:  http://www.gamerzplanet.net/forums/asm/197748-tut-asm-
beginners-advanced-users.html
```

5.

```
;programa que pateticamente muestra un descansador de pantalla, el cual funciona
;solamente la primera impresion lo demas saka basura hay que corregirlo!!!
```

6.

```
; programa que cambia de minusculas a mayusculas (Y)
```

7.

```
; mitad del programa del editor de texto, aka se fuma un proceso interesante
; que deja ir metiendo askis sin necesidad de una cadena sin verificacion, lo
que permite
; que se cuenten la cantidad de caracteres que se kieren por linea y ademas
contiene una
; verificacion de cuantos enters puede hacer muy bueno
```

8.

```
 ; programa editor de texto mas avanzado que permite crear archivos en una
 locacion
 ; y abrir archivos de texto en distintas paginas de memoria
```

9.

```
PROGRAMA EL CUAL REALIZA CUATRO PORCESOS SIMULTANEOS
```

10.

```
; editor de texto mas avanzado con teclas y funciones especiales.
```

11.

```
;Ejercicio No.3
;Programa que contien unmenu contenido en tablas
;tiene 4 opciones para realizar siendo la ultima
;de salida
```

```
; Nery Guzman
;2007 Universidad del Valle de Guatemala
;http://docs.huihoo.com/help-pc/ pagina de referencia donde estan las
interrupciones

;EXCEPCION AH07 de la int 21 h
;programa que lee una tecla del buffer sin repeticion en pantalla( sin eco)
; luego se manda con la excepcion ah07 dentro de al y se compara
; siendo 00 tecla de funcion
.MODEL SMALL
.STACK 64


.DATA

.CODE



JUAN PROC FAR



MOV AH, 07H       ; Petición de la función
      INT 21H                 ; lee la tecla, no hace eco
      CMP AL, 00        ; en AL retorna carácter leído.
                              ; Es 00 (tecla de función)?
      JNZ SALIDA        ; si NO es 00, termina. Ya está carácter en AL

SALIDA: MOV AH,4CH
            INT 21H


JUAN ENDP

END JUAN



title lee los sectores de disco
.model small
.stack 64
;-------------------------------------------------------------------------
.data
row         db    00
col         db    00
xlatab      db      30h,31h,32h,33h,34h,35h,36h,37h,38h,39h
            db      41h,42h,43h,44h,45h,46h
readmsg     db    '*** read error ***', 0dh, 0ah

rdblock     db    0           ; estructura del bloque
rdhead      dw    0           ; cabeza lectura/escritura
rdcylr      dw    0           ; cilindro
rdsect      dw    4           ; sector inicial
rdnosec     dw    10          ; numero de sectores
rdbuffr     dw    iobuffr     ; desplazamiento de buffer de entrada,
                              ; proporciona un sector de datos
            dw    seg _data   ; operador seg identifica direccion del
                              ; segmento de datos
iobuffr     db      5120 dup (' ')
mensaje db ('ingrese el sector abs '), '$'
entrada     db    20 dup (`  `)     ; área de entrada
```

```
buffer db 0 , '$'
blur dw 0
arreglo           db    80 dup (?),'$'   ;área de caracteres a copiar
pagina db 0
cursor db 0
x dw 4
y db 0
residuo           db          ?
;-------------------------------------------------------------------
.code


sectorA proc near
      xor ax, ax              ; inicializar ax
      mov al, buffer          ; cargo lo quiero dividir en ax
      mov bl, 80              ; cargo el divisor a bl
      div bl                        ; se divide
      mov rdcylr, ax    ; se mueve el coiente a cilindro
      mov residuo, ah         ; se mueve el ah a resudio

      xor ax, ax              ; inicializar ax
      mov al, residuo
      mov bl, 18
      div bl
      mov rdsect, ax    ; muevo el ah a sectores



sectorA endp
conversion1 proc far
mil:
      mov x,4
      mov bx,0
      mov al, [es:di]
      mov al,arreglo[bx]
      sub x,bx
multi2:
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      dec x
      cmp x,0
      jne multi2

cien:
      mov x,4
      mov bx, 1
      mov al,arreglo[bx]
      sub x,bx
multi3:
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
```

```
      add buffer, al
      add buffer, al
      add buffer, al
      dec x
      cmp x,0
      jne multi3
diez:
   mov x, 4
      mov bx, 2
      mov al,arreglo[bx]
      sub x,bx

multi:
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      add buffer, al
      dec x
      cmp x,0
      jne multi

uni:
      mov al, arreglo[3]
      add buffer, al

mov ah, 09h
lea dx, buffer
int 21h




conversion1 endp
getcursor proc near

                              ;llama a getpagina para posicionarse en
      mov ah, 03h                         ;pagina actual
      mov bh, 0h
      int 10h
      mov ax, dx
      mov cursor, dh
      ret

getcursor endp

pedir proc far
mov ah, 09h
lea dx, mensaje
int 21h

ciclus:

mov ah, 07h
int 21h
mov buffer, al
cmp al, 0dh

mov bx,blur
```

```
      mov al, [es:di]
      mov al, buffer
      mov arreglo[bx], al
      inc blur


mov ah, 0ah     ;se llama al modo ingresar caracter
      mov al, buffer
      mov bh, 0h;se pone la pagina actual
      mov cx, 01        ;se dice cuantas veces se va a imprimir en la pocision
del cursor esa letra
      int 10h           ;interrupcion

      call getcursor    ;se llaman las coordenadas del cursor para avanzar
      add dl, 01h       ;se avanza un espacio el cursor
      mov ah, 02h       ;petición
      mov bh, 0h  ;se pone la pagina
      int 10h
cmp blur, 4
jne ciclus

ret
pedir endp

main  proc far
      mov ax, @data           ; inicializa registros de segmento
      mov ds, ax
      mov es, ax
      call pedir
      call conversion1
      call limpia_pantalla    ; limpia pantalla
      call pos_cursor         ; coloca el cursor
      call lee_sector         ; obtiene datos del sector

      jnc continuar           ; si lectura es valida, continuar.
      lea dx, readmsg         ; lectura no valida
      call error
      jmp salida

continuar:
      call despliegue         ; convertir y desplegar
salida:
      mov ax, 4c00h           ; salir al dos
      int 21h
main  endp
;---------------------------------------------------------------
;     recorrido de la pantalla
limpia_pantalla   proc near
      mov ax, 0600h     ; peticion
      mov bh, 1eh       ; atributo
      mov cx, 0000
      mov dx, 184fh
      int 10h
      ret
limpia_pantalla   endp
;---------------------------------------------------------------
;     establece cursor
pos_cursor proc near
        mov ah, 02h            ; peticion
        mov bh, 00
        mov dh, row            ; renglon (fila)
        mov dl, col            ; columna
        int 10h
        ret
```

```
pos_cursor endp
;---------------------------------------------------------------
;     lee los datos del sector
lee_sector  proc near
      mov ax, 440dh     ; ioctl para dispositivo de bloque
      mov bx, 01        ; 0=unidad por default, 1=unidad a, 2=unidad c
      mov ch, 08        ; categoria del dispositivo
      mov cl, 61h       ; lee sector
      lea dx, rdblock   ; direccion para estructura de bloque
      int 21h
      call sectora
      ret
lee_sector  endp
;---------------------------------------------------------------
;       desplegar mensaje de error en disco
error  proc near
mov ah, 40h     ; desplegar mensaje, dx contiene direccion        mov bx, 01
; manejador 01 = pantalla
        mov cx, 20      ; longitud de mensaje
        int 21h
        inc row
        ret
error  endp
;---------------------------------------------------------------
;     desplegar datos del sector
despliegue  proc near
        lea si, iobuffr
c20:
      mov al, [si]
      shr al, 1
      shr al, 1
      shr al, 1         ; shr al, 4 equivale pero da error
      shr al, 1         ; correr a la derecha un digito hex

      lea bx, xlatab    ; designar direccion a la tabla
      xlat              ; traducir el hex
      call caracter
      inc col
      mov al, [si]
      and al,0fh        ; borrar digito hex de la izq
      xlat              ; traducir el hex
      call caracter
      inc si
      inc col
      cmp col, 64
      jbe c20
      inc row
      mov col, 00
      call pos_cursor
      cmp row, 16
      jbe c20
      ret
despliegue  endp
;---------------------------------------------------------------
caracter proc near
        mov ah, 02h             ; peticion para imprimir caracter
        mov dl, al
        int 21h
        ret
caracter endp
;---------------------------------------------------------------
        end main
```

```
;Universidad del Valle de Guatemala
;Nery Fernando Guzman Carn, 06153
;Luis Fernando Reina  Carn, 05070
;http://docs.huihoo.com/help-pc/ pagina de referencia donde estan las
interrupciones
; Programa Tarea NO. 1 usa como base Programa Ejercio No. 1 Temario No. 3
;Sumar los numeros pares de cinco n£meros predefinidos, menores que 100
; y mayores que 0. La suma debe tener un resultado de 2 digitos.
;Desplegar el Resultado, el cual debe de ser menor que 100.


.MODEL SMALL
.STACK 64
.DATA
;-------------------------------------------------------------------------
; Definicion de datos
;-------------------------------------------------------------------------
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; Para el manejo de las cadenas.
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
NUMERO            LABEL BYTE        ; Inicio de lista de parametros
MAXLEN            DB    06    ; Longitud máxima (recibe 6 digitos y enter)
ACTLEN            DB    ?           ; Longitud ingresada
DATOS       DB    06 DUP ('.')      ; Datos ingresados en teclado
;-------------------------------------------------------------------------
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; Definicion de constantes, arreglos y cadenas.
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
NUM         DW    05 DUP (0)  ;Guarda los n£meros ingresados
NUM_6       DW      00        ;Guarda la suma de los pares
Cent        DW    ?           ;Guarda centenas
Dece              DW      ?         ;Guarda decenas
Unid        DW      ?         ;Guarda unidades
Salida            DB    03 DUP (';')      ;Guarda la salida
TERMI       DB    '$'         ;Terminador para la salida
CADENA      DB    13,'Listados : ','$'
INGRES            DB    10,13,'Por favor ingrese un numero: ','$'
BIENVE            DB    'Ingrese 5 némeros.',10,13,'$'
conta       db  05
;-------------------------------------------------------------------------


;-------------------------------------------------------------------------
; Inicio del codigo
;-------------------------------------------------------------------------
.CODE
MAIN  PROC FAR

      MOV AX,@data      ;inicializa segmento de datos
      MOV DS,AX

      MOV AL, 0H

      MOV AH, 09h             ;muestra mensaje en pantalla
    LEA DX, BIENVE
            INT 21H

      CALL Contar       ;Pide y suma los números.

      MOV AH, 0AH       ; Peticion
      LEA DX, NUM_6           ; Espera un enter para salir al DOS
      INT 21H
```

```
            MOV AH, 4CH             ;salida al DOS
            INT 21H

MAIN    ENDP

;-------------------------------------------------------------------------
;       Procedimiento Par
;-------------------------------------------------------------------------
Par proc far

      MOV NUM_6, 00H          ; Se borra NUM_6 que contiene la suma de numeros
pares
      MOV BX, 10        ; Contador inicial a 10
NEXT: MOV AX, NUM[BX-2] ; Se trabaja de 2 en 2 por ser una palabra y no bytes
      AND AL, 0001H           ; Se hace la mascara en BX
      JPO impar               ; Bit de paridad define el salto
            MOV AX, NUM[BX-2]
      ADD AX, 0h            ; Suma el num6 a AX
            MOV NUM_6, AX           ; Mueve a num6 el AX (acumulador)
impar:      SUB BX, 2         ; Resta 2 por ser palabra
      JNZ NEXT
            RET                     ; retorna al programa
Par endp
;-------------------------------------------------------------------------
;       Procedimiento Pedir
;-------------------------------------------------------------------------
Pedir proc far

      MOV AH, 09h       ;muestra mensaje en pantalla
      LEA DX, INGRES
      INT 21H

      MOV AH, 0AH       ; Peticion
      LEA DX, NUMERO          ; Carga direccion de la
      INT 21H                 ; lista de parametros
      RET
Pedir endp
;-------------------------------------------------------------------------
;       Procedimiento Conver
;-------------------------------------------------------------------------
Conver      proc far

      MOV BX, 3         ; Inicia a 0 BX
SIG:  MOV AL, DATOS[BX-1]     ; Mueve a AL el los datos ingresados LIFO
      SUB AL, 30H       ; Resta 30H para quitar el ASCII
      MOV DATOS[BX-1], AL     ; MUeve el nuevo dato al registro
      DEC BX                  ; Resta 1 al contador
      JNZ SIG                 ; Salta si no est  en 0.
      RET               ; Regresa a ejecuci¢n
Conver      endp
;-------------------------------------------------------------------------
;       Procedimiento Contar
;-------------------------------------------------------------------------
Contar      proc far

      MOV SI, 10
Snum: Call Pedir        ; Pide un numero
      CALL Conver       ; Convierte los numeros ascii a numeros
      CALL Suma               ; Guarda la suma en una constante.
      MOV NUM[SI-2], AX ; Guarda un numero

      ;************************************
      ;la suma de los numeros esta en NUM_6
      ;************************************
```

```
      CALL Par          ; Comprueba que numeros son pares y los suma
            ; Despliega en pantalla la suma de los pares hasta el momento.

      SUB SI, 2         ; Resta 2 al contador por trabajar con palabras
      JNZ Snum          ; Regresa para pedir el siguiente numero

      RET
Contar      endp
;--------------------------------------------------------------------------
;       Procedimiento Suma
;--------------------------------------------------------------------------
Suma  proc far

      MOV AX, 0000H           ; Inicia a 0 al


      mov bx, 5h
      MOV AL, DATOS[bx-1]     ; Guarda en AL el numero actual en datos
      MOV CX, 100       ; Multiplica por 100
      MUL CX
      MOV CENT, AX            ; guarda en centenas
      MOV AX, 00H       ; Inicia a 0 al
      MOV AL, DATOS[1]  ; Guarda en AL el numero actual en datos
      MOV CX, 10        ; Multiplica por 10
      MUL CX
      MOV DECE, AX            ; Guarda en decenas
      MOV AX, 00H       ; Inicia a 0 al
      MOV AL, DATOS[2]  ; Guarda en AL el numero actual en datos
      MOV CX, DECE            ; mueve decenas a cl
      ADD AX, CX        ; suma decenas y unidades
      MOV CX, CENT            ; mueve centenas a cl
      ADD AX, CX        ; suma centenas a las decenas y unidades
      RET               ; Regresa (deja en AL la suma)

Suma  endp
;--------------------------------------------------------------------------
        ;------------------------------------------------------------------------
-
;       Procedimiento Desplegar
;--------------------------------------------------------------------------
Dsplgr      proc far

            MOV AH, 09h             ;muestra mensaje en pantalla
      LEA DX, CADENA
            INT 21H
                              ;debemos convertir el numero para poder mostrarlo
correctamente
            MOV AX, NUM_6           ;Se coloca el n£mero a dividir en registro
AX
            MOV BL, 100             ;En registro BL se coloca el divisor
            DIV BL                  ;Se divide
            MOV SALIDA[0], AL       ;Cociente se obtiene en AL (Centenas)
            MOV SALIDA[1], AH       ;Residuo se obtiene en AH  (Decenas y
unidades)

      AND AX, 0000H           ;Se aplica una mascara para borrar ax
            MOV AL, SALIDA[1]       ;Se coloca el n£mero a dividir en registro
AX
            MOV BL, 10              ;En registro BL se coloca el divisor
            DIV BL                  ;Se divide
            MOV SALIDA[1], AL       ;Cociente se obtiene en AL (Decenas)
            MOV SALIDA[2], AH       ;Residuo se obtiene en AH  (Unidades)
```

```
            ADD SALIDA[0], 30H      ;Se suma 30H para desplegar ASCII
      ADD SALIDA[1], 30H      ;Se suma 30H para desplegar ASCII
            ADD SALIDA[2], 30H      ;Se suma 30H para desplegar ASCII

      MOV DX, 00H             ;Se borra DX
            LEA DX, SALIDA[0]       ;Se coloca en DX la direcci¢n de Centenas

            MOV AH, 09H             ;muestra el caracter en pantalla
            INT 21H
            RET
Dsplgr endp

            END MAIN
```

```
; Nery Guzman
;2007 Universidad del Valle de Guatemala
;http://docs.huihoo.com/help-pc/ pagina de referencia donde estan las
interrupciones

;EXCEPCION AH07
; programa que hace 2cuadros en el centro de la pantalla, posiciona el cursor y
deja escribir 20 caracteres sin verificar
; por  medio del manejador de teclado
.MODEL SMALL
.STACK 64


.DATA

ENTRADA     DB    20 DUP (‘  ‘)     ; Área de entrada
ENTRADA2    DB    20 DUP (‘  ‘)     ; Área de entrada2
.CODE



eje   PROC FAR

.startup

; se crea el primer cuadro

MOV AX, 0605H
MOV BH, 61H       ;
MOV CX, 0A0CH           ; Desde fila 10, columna 28
MOV DX, 0E34H           ; Hasta fila 14, columna 52
INT 10H

; se posiciona el cursor
MOV AH, 02H       ; Petición
MOV BH, 0         ; Número de página
MOV DH, 0Ah       ; Fila 10
MOV DL, 12        ; Columna 12 (MOV DX, 050CH)
INT 10H


; se usa el manejador con la primera cadena
MOV AH, 3FH             ; Petición
MOV BX, 00              ; Manejador para teclado
MOV CX, 20              ; Máximo 20 caracteres
LEA DX, ENTRADA         ; Carga la dirección del área de
; entrada


INT 21H                       ; Llama al DOS


;se cambia de pagina de memoria

MOV AH, 05H       ; Petición
MOV AL, 01h ; Número de página
INT 10H



; se hace el 2ndo cuadro
MOV AX, 0605H
MOV BH, 46H       ;
MOV CX, 0A0CH           ; Desde fila 10, columna 28
```

```
MOV DX, 0E34H           ; Hasta fila 14, columna 52
INT 10H

;se osiciona el cursor
MOV AH, 02H       ; Petición
MOV BH, 1         ; Número de página
MOV DH, 0Ah       ; Fila 10
MOV DL, 12        ; Columna 12 (MOV DX, 050CH)
INT 10H

mov al,00h


;se manejan otros 20 caracteres
MOV AH, 3FH             ; Petición
MOV BX, 00              ; Manejador para teclado
MOV CX, 20              ; Máximo 20 caracteres
LEA DX, ENTRADA2        ; Carga la dirección del área de
                                  ; entrada
INT 21H                       ; Llama al DOS

mov ax,0



      MOV AH, 4CH             ;salida al DOS
    INT 21H


eje endp

end eje
```

```
; Nery Guzman
;2007 Universidad del Valle de Guatemala
;http://docs.huihoo.com/help-pc/ pagina de referencia donde estan las
interrupciones


;este programa lo que hace es agarrar un arreglo que ingresa el usuario y le
quita todos los espacios en blanco
; y lo imprime de nuevo sin los espacios ( aca se debio haber usado scas y stows
pero yo utilize escritura directa en memoria
; para explorar y reescribir los arreglos)


buskeda macro
add contador,01h
      mov al,contador
      mov ah, 02h             ;petición
      mov bh, pagina          ;número de página
      mov dh, contador  ;fila x
      mov dl, 0               ;columna 0 para empezar la linea
      int 10h
      mov ah,10h
      int 16h

      mov blur, 0h

sigue2:
      mov bx, blur
      inc blur
      cmp al, arreglo[bx]
      jne sigue2
      inc letra
      cmp bx,40h
      jb sigue2
endm

conversion macro

; ah= residuo
; al = cociente



endm

.model small
.stack 64
;-------------------------------------------------------------------------
.data                                    ;definicion de datos
entrada           db          3 dup ()    ;área de entrada
arreglo           db    80 dup (?),'$'   ;área de caracteres a copiar
contador    db          0
cont        db          0
aux         dw          0
aux2  db    0
aski        db          0
pagina      db          0
cursor            db          0
kursor            dw          0
diez    db  10

blur       dw       0
letra dw 0,'$'
residuo           db          ?
```

```
comparacion db 0
;-------------------------------------------------------------------------
.code

;-------------------------------------------------------------------------
backspace proc near

      call getcursor ;se mandan a traer las coordenadas del cursor
      cmp dl,0h
      je dont
      add dl,-01h       ;se atraza un espacio el cursor
      mov ah, 02h       ;petición
      mov bh, pagina    ;se pone la pagina
      int 10h                 ;avanza uno el cursor para simular el avanze de un
space despues de un tecladazo
      mov aski,20h
      call getpagina    ;se llama las coordenadas de la pagina
      mov ah, 0ah     ;se llama al modo ingresar caracter
      mov al, aski      ;se guarda el aski metido anteriormente en la variable
aski
      mov bh, pagina  ;se pone la pagina actual
      mov cx, 01        ;se dice cuantas veces se va a imprimir en la pocision
del cursor esa letra
      int 10h           ;interrupcion

      mov ah, 02h       ;petición
      mov bh, pagina    ;se pone la pagina
      int 10h                 ;se atrasa 2 veces porque el metodo ascii avanza
uno entonces solo hubiera dejado un espacio en blanco DX
dont:
      ret

backspace endp
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
getpagina proc near

      mov ah, 0fh
      int 10h                                   ;se guarda en bh el numero de
pagina actual
      mov pagina, bh
      ret

getpagina endp
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
getcursor proc near

      call getpagina                    ;llama a getpagina para posicionarse
en
      mov ah, 03h                       ;pagina actual
      mov bh, pagina
      int 10h
      mov ax, dx
      mov cursor, dh
      ret

getcursor endp
;-------------------------------------------------------------------------
;proceso ascii es el que escribe la letra en pantalla
;-------------------------------------------------------------------------
ascii proc near

      call getpagina    ;se llama las coordenadas de la pagina
```

```
      mov ah, 0ah     ;se llama al modo ingresar caracter
      mov al, aski      ;se guarda el aski metido anteriormente en la variable
aski
      mov bh, pagina  ;se pone la pagina actual
      mov cx, 01        ;se dice cuantas veces se va a imprimir en la pocision
del cursor esa letra
      int 10h           ;interrupcion

      cmp aski,20h ;SE COMPARA SI ES BARRA ESPACIADORA
      je sigue  ; SI ES BARRA ESPACIADORA NO SE COPIA EN LA CADENA ARREGLO
      call getcursor
      ;ACA SE VA METIENDO EL ASCII DE MEMORIA AL ARREGLO
      mov bx,blur

      mov al, [es:di] ;esto es lo que reemplaza a las funcionas scas
      mov al, aski
      mov arreglo[bx], al
      inc blur
sigue:
      inc aux           ;se incrementa la linea
      inc aux2


      pop bx
      call getcursor    ;se llaman las coordenadas del cursor para avanzar
      add dl, 01h       ;se avanza un espacio el cursor
      mov ah, 02h       ;petición
      mov bh, pagina    ;se pone la pagina
      int 10h                 ;avanza uno el cursor para simular el avanze de un
space despues de un tecladazo
      ret



ascii endp
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
insert proc far
      ; esto hace la peticion para escribir 80 caracteres por linea
      ; para con enter y llena de espacios

ingreso:
      mov ah,10h
      int 16h

      mov aski, al
      cmp ax, 1c0dh     ;es enter ??
      je clinea         ;si si
      cmp ax, 0e08h   ;es backspace?
      je cback          ;si si brinka a backspace :P

      call ascii        ;si no es tecla especial brinca al asciii

      cmp aux, 50h      ;compara si ya llego al final de la linea adimitido
      ja clinea         ;si ya llego entonces brinca a cambio de linea simula
enter
      jmp ingreso         ;si no simula de nuevo la entrada para seguir metiendo
askis

clinea:

      call c2linea
      ret
```

```
cback:
      call getpagina
      call backspace
      ret



insert endp
;-------------------------------------------------------------------------
c2linea proc far



      call getpagina          ;se llama la pagina actual
                                    ;proceso para cambiar de linea despues de
que ya termino de escribir
      add contador,01h
      mov al,contador
      mov ah, 02h             ;petición
      mov bh, pagina          ;número de página
      mov dh, contador  ;fila x
      mov dl, 0               ;columna 0 para empezar la linea
      int 10h
      mov aux,0h
      call getcursor
      mov kursor,dx
      MOV AH, 09h       ;muestra mensaje en pantalla
      LEA DX, arreglo
      INT 21H
      call buskeda2

      ret


c2linea endp
;-------------------------------------------------------------------------
buskeda2 proc far


      call getpagina          ;se llama la pagina actual
                                    ;proceso para cambiar de linea despues de
que ya termino de escribir

      buskeda
      call exit
ret




buskeda2 endp

exit proc far
      dec letra
      mov ax, letra
      cmp ah,0
      je fuera
      div diez



fuera:
```

```
      add letra, 30h
      MOV AH, 09h       ;muestra mensaje en pantalla
      LEA DX, letra
      INT 21H
      mov ah,10h
      int 16h
      mov ah, 05h ; salida de protocolo DX
      mov al, 0
      int 10h
      mov ax, 0600h
      mov bh, 07h
      mov cx, 0000
      mov dx, 184fh
      int 10h
      mov ax,4c00h       ;salida al dos
      int 21h

      ret

exit endp
begin proc far

      mov ax, @data   ; inicializar area de datos
      mov ds, ax
      mov ah, 00h     ; cargo a ah 00h
      mov al, 03h     ; asigna modo texto a color 80x25
      int 10h
insertado:
      call insert ;llama al procedimiento q empieza a dibujar

        cmp contador, 5
            jne insertado

      MOV AH, 09h       ;muestra mensaje en pantalla
        LEA DX, arreglo
        INT 21H

      mov ah,10h
      int 16h


salida:


      mov ah, 05h ; salida de protocolo DX
      mov al, 0
      int 10h
      mov ax, 0600h
      mov bh, 07h
      mov cx, 0000
      mov dx, 184fh
      int 10h
      mov ax,4c00h       ;salida al dos
      int 21h
      ret
begin   endp
end begin
;-------------------------------------------------------------------------
```

```
;File Writing by 414C485649
;     Escribe un arcrivo en el disco utilizando las funciones que provee el DOS
con la int 21h
;Info y ejemplo:  http://www.gamerzplanet.net/forums/asm/197748-tut-asm-
beginners-advanced-users.html
.model small
.stack 64
;-------------------------------------------------
.data
fileName          db          'fileASMTest.txt$'
fileDescriptor    dw          ?
dataToWrite       db          'Line 1', 0dh, 0ah, 'Line 2', '$'
errorCreating     db          'Error al crear el archivo', 0dh, 0ah, '$'
errorWriting      db          'Error al escribir en el archivo', 0dh, 0ah,'$'
success                 db          'Archivo creado con exito', 0dh, 0ah, '$'
pressAKey         db          'Presione una tecla para salir $'
;-------------------------------------------------------------
.code
main  proc far
      .startup

      call createFile
      jc cError
      call writeFile
      jc wError
      lea dx, success
      call cout
      jmp endProgram
      cError:
            lea dx, errorCreating
            call cout
            jmp endProgram
      wError:
            lea dx, errorWriting
            call cout
      endProgram:
            lea dx, pressAKey
            call cout
            call cin
            mov ax, 4c00h                ; salir al DOS
            int 21h
main  endp
;-------------------------------------------------------------
createFile proc near
      mov ax, 3c00h                      ; Crear archivo
      xor cx, cx                         ; cx = 0, atributo normal
      lea dx, fileName              ; nombre del archivo
      int 21h
      mov fileDescriptor, ax        ;guarda el handler
      ret
createFile endp
;-------------------------------------------------------------

;-------------------------------------------------------------
writeFile proc near
      mov ax, 4000h                      ;Escribir a archivo
      mov bx, fileDescriptor
      mov cx, 14                         ; Numero de bytes a escribir
      lea dx, dataToWrite                ;data
      int 21h
      ret
writeFile endp
;-------------------------------------------------------------
```

```
cout proc near
      mov ax, 0900h
      int 21h
      ret
cout endp
;-------------------------------------------------------------
cin proc near
      mov ax, 0a00h
      int 21h
      ret
cin endp
;-------------------------------------------------------------
end main
```

```
; Nery Guzman
;2007 Universidad del Valle de Guatemala
;http://docs.huihoo.com/help-pc/ pagina de referencia donde estan las
interrupciones

;programa que pateticamente muestra un descansador de pantalla, el cual funciona
;solamente la primera impresion lo demas saka basura hay que corregirlo!!!

TITLE

.MODEL SMALL
.STACK 64
.DATA



; Definicion de datos
data db ' ****
'
     db ' ****'
data2 db' ++++
'
     db ' ----'
data23 db ' nery
'
     db ' Dman'
up dw 0
tope dw 85



.CODE

insert proc far



insert endp

;-------------------------------------------------------------------------------
---------
exit proc near



      MOV AH, 05H ; Petición
      MOV AL, 0   ; Número de página
      INT 10H

      mov ax,4C00H       ;salida al DOS
      INT 21H


exit endp
;-------------------------------------------------------------------------------
-----------


start proc near

      mov ah, 07H ;garbage
      int 21H
      mov dx, tope
      MOV AH, 05H ; Petición
```

```
      MOV AL, 00  ; Número de página
      INT 10H

      mov ax,0B800h   ; direccion de inicio de memoria de video
    mov es,ax       ; se carga al registro ES la dir direcc mem
      mov di,0


tag1:   mov up, 0     ; inicializa contador a cero
        lea bx, data    ; mensaje se carga en bx
        mov ah,5h    ; atributo



ciclus:     mov al,[bx]
        mov ah, 6h



STOSW


            cmp di,320
            jne etiketaloka1
            mov di, 0


etiketaloka1:
            inc bx          ; incrementa puntero de la cadena
        inc up       ; incrementa contador
        cmp up, dx    ; hasta llegar al tamaño de la cadena
        jne ciclus

            MOV ah, 01h ;
            INT 16h           ; REPETIR
            JNZ COMP    ; CLICLO
            JMP tag1          ; HASTA Q haya tecladazo


comp: jmp ciclus


            mov   ah, 00h
            int 16h
            CMP AL, 0dH ; COMPROBAR SI ES ENTER
            Je gogogo   ;
            mov di,0
            CALL exit   ; si no es enter se sale

gogogo:     CALL pag2   ; si es enter pasa


start endp
;-------------------------------------------------------------------------------
-----------

pag2 proc near

    mov di,0
      mov ah, 07H ; interrupcion para limpiar basura si no se hace zaz
      int 21H
    mov dx, tope
      MOV AH, 05H ; Petición
      MOV AL, 01  ; Número de página
```

```
      INT 10H

      mov ax,0B900h   ; direccion de inicio de memoria de video
    mov es,ax       ; se carga al registro ES la dir direcc mem
      mov di,168


tag12:   mov up, 0     ; inicializa contador a cero
        lea bx, data2   ; mensaje se carga en bx
        mov ah,5h    ; atributo inicial
            mov dx, tope
            add di, -168
ciclus2:    mov al,[bx]
        mov ah, 6h



STOSW
cmp di,320
jne etiketaloka2
mov di, 168
etiketaloka2:
            inc bx          ; incrementa puntero de la cadena
        inc up       ; incrementa contador
        cmp up, dx    ; hasta llegar al tamaño de la cadena
        jne ciclus2

            MOV ah, 01h ;
      INT 16h
      JNZ COMP2
      JMP tag12         ; esperar tecladazo
      comp2: jmp ciclus2


      mov   ah, 00h
      int 16h
      CMP AL, 0dh ; enter
      Je gogogo2  ;
      CALL exit   ;si no se sale

gogogo2:    CALL pag3   ; si si continua

pag2 endp


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

pag3 proc near



      mov ah, 07H ; interrupcion para limpiar basura si no se hace zaz
      int 21H
    mov dx, tope
      MOV AH, 05H ; Petición
      MOV AL, 03  ; Número de página
      INT 10H

      mov ax,0Ba00h   ; direccion de inicio de memoria de video
    mov es,ax       ; se carga al registro ES la dir direcc mem
      mov di,168
```

```
tag123:   mov up, 0     ; inicializa contador a cero
        lea bx, data23  ; mensaje se carga en bx
        mov ah,5h    ; atributo inicial
            mov dx, tope
            add di, -168
ciclus23:   mov al,[bx]
        mov ah, 6h



STOSW
cmp di,320
jne etiketaloka23
mov di, 168
etiketaloka23:
            inc bx          ; incrementa puntero de la cadena
        inc up       ; incrementa contador
        cmp up, dx    ; hasta llegar al tamaño de la cadena
        jne ciclus23

            MOV ah, 01h ;
      INT 16h           ; REPETIR
      JNZ COMP23  ; CLICLO
      JMP tag123        ; HASTA QUE EL USUARIO PRESIONE
comp23: jmp ciclus23


      mov   ah, 00h
      int 16h
      CMP AL, 0DH ;ver si es enter
      jz gogogo23 ;
      CALL exit   ; si no es enter se sale

gogogo23:   CALL start  ; regresa al principio


pag3 endp

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

begin proc far


        mov ax, @DATA   ; inicializar area de datos
            mov ds, ax
        mov ah, 00h     ; cargo a ah 00h
        mov al, 03h     ; asigna modo texto a color 80x25
        int 10h

      CALL start  ;llama al procedimiento q empieza a dibujar

begin   endp
      end begin


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

```
; Nery Guzman
;2007 Universidad del Valle de Guatemala
;http://docs.huihoo.com/help-pc/ pagina de referencia donde estan las
interrupciones

; programa que cambia de minusculas a mayusculas (Y)


TITLE MINMAY Conversion de minusculas a mayusculas
;-------------------------------------------------
; las letras de la A a la Z son desde 41H a 5AH
; las letras de la a a la z son desde 61H a 7AH
; la diferencia se encuentra en el bit 5: mayusc 0, minusc 1

.MODEL SMALL
.STACK 64
.DATA
; Definicion de datos


tiket DB    ?                       ; numero real de caracteres de entrada


;-------------------------------------------------
; Inicio del codigo
.CODE




print proc far

            mov ax,0B800h ; dirección de inicio de memoria de video
                          ; pagina 0
        mov es,ax          ; se carga al registro ES la direcc mem ¡MUY
IMPORTANTE ¡



            cmp bl,30h
            je cero



            mov di,0
        mov cx,2000       ; numero de veces que se repite la impresion
        mov al,31h        ; se carga el caracter que va a imprimirse (B)
        mov ah,5           ; atributo
        rep stosw           ; almacena el contenido del acumulador en
                                 ; la memoria. El prefijo REP junto con CX
                                 ; hace que se repita la operacion CX veces
                                 ; El par ES:DI hace referencia al area de
                                 ; memoria donde sera almacenado

            mov ax,4C00H
            int 21h

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

            mov ax,0B900H ; dirección de inicio de memoria de video
                          ; pagina 1
        mov es,ax          ; se carga al registro ES la direcc mem ¡MUY
```

```
IMPORTANTE ¡



            cmp bl,30h
            je cero



            mov di,0
          mov cx,2000      ; numero de veces que se repite la impresion
          mov al,31h       ; se carga el caracter que va a imprimirse (B)
          mov ah,5          ; atributo
          rep stosw          ; almacena el contenido del acumulador en
                                  ; la memoria. El prefijo REP junto con CX
                                  ; hace que se repita la operacion CX veces
                                  ; El par ES:DI hace referencia al area de
                                  ; memoria donde sera almacenado

            mov ax,4C00H
            int 21h



            jmp enddd

cero:
            mov di,0
          mov cx,2000      ; numero de veces que se repite la impresion
          mov al,30h       ; se carga el caracter que va a imprimirse (B)
          mov ah,5          ; atributo
          rep stosw          ; almacena el contenido del acumulador en
                                  ; la memoria. El prefijo REP junto con CX
                                  ; hace que se repita la operacion CX veces
                                  ; El par ES:DI hace referencia al area de
                                  ; memoria donde sera almacenado
            mov ax,4C00H
            int 21h


;-----------------------------------------------------------------
            166
            2
            mov ax,0B900H ; dirección de inicio de memoria de video
                          ; pagina 1
          mov es,ax         ; se carga al registro ES la direcc mem ¡MUY
IMPORTANTE ¡

            mov di,0
          mov cx,2000      ; numero de veces que se repite la impresion
          mov al,30h       ; se carga el caracter que va a imprimirse (B)
          mov ah,7          ; atributo
          rep stosw          ; almacena el contenido del acumulador en
                                  ; la memoria. El prefijo REP junto con CX
                                  ; hace que se repita la operacion CX veces
                                  ; El par ES:DI hace referencia al area de
                                  ; memoria donde sera almacenado
            mov ax,4C00H
            int 21h



enddd:  MOV AH, 4CH      ;salida al DOS
```

```
            INT 21H

print endp

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

BEGIN PROC FAR

        MOV AX,@data     ;inicializa segmento de datos
            MOV DS,AX


;Limpiar la pantalla



MOV AX, 0600H           ; Petición  AH=06, AL 00 pantalla completa
MOV BH, 71H       ; atributo blanco (7) sobre azul (1)
MOV CX, 0000H     ; esquina superior izq fila: columna.
; Desde 00,00
MOV DX, 184FH           ; esquina inferior der fila: columna
                        ; Hasta 24, 79 (pantalla completa)
INT 10H

; peticion del teclado

MOV AH, 07H       ; Petición de la función
INT 21H                 ; lee la tecla
mov bl, al

;llenar la pantalla de letra loka
call print



BEGIN ENDP
;-------------------------------------------------
END BEGIN
```

```
; Nery Guzman
;2007 Universidad del Valle de Guatemala
;http://docs.huihoo.com/help-pc/ pagina de referencia donde estan las
interrupciones

; mitad del programa del editor de texto, aka se fuma un proceso interesante
; que deja ir metiendo askis sin necesidad de una cadena sin verificacion, lo
que permite
; que se cuenten la cantidad de caracteres que se kieren por linea y ademas
contiene una
; verificacion de cuantos enters puede hacer muy bueno

TITLE

.MODEL SMALL
.STACK 64
.DATA



; DEFINICION DE DATOS

ENTRADA     DB    40 DUP (`  `)     ; ÁREA DE ENTRADA
COPI    DB  40 DUP (`  `)
CONTADOR DB 0
CONT DB 0
tabulador db '    ','$'
fila db 0
aux db 0
loc db 0
renglon db ?
pagina db ?

.CODE
;-------------------------------------------------------------------------------
-----------
doend proc far
mov ah,02h
mov bh,00
mov dx, 184fh
int 10h
doend endp
;-------------------------------------------------------------------------------
-----------
getpagina proc far
mov ah,0fh
int 10h  ; se guarda en bh el numero de pagina actual XD
mov pagina, bh
ret
getpagina endp
;-------------------------------------------------------------------------------
----------s
getcursor proc far
mov ah,03h
mov bh,00
int 10h
mov ax,dx

ret
getcursor endp
;-------------------------------------------------------------------------------
-----------
home proc far
CALL getpagina
```

```
add dh,-1
;call backspace
MOV AH, 02H       ; PETICIÓN
MOV BH, PAGINA          ; NÚMERO DE PÁGINA
MOV DH, 00H       ; FILA 0
MOV DL, 0H        ; COLUMNA 0 (MOV DX, 050CH)
INT 10H
mov aux,0h
mov contador,0h
ret

home endp

;-------------------------------------------------------------------------------
-----------
tab proc far
call getcursor

MOV AH, 02H       ; PETICIÓN
INT 10H

add aux,07h
ret
tab endp
;-------------------------------------------------------------------------------
-----------

COPY PROC FAR
call getcursor


RET

COPY ENDP
;-------------------------------------------------------------------------------
-----------
;PROCESO HECHIZO PARA SALIR :P
EXIT PROC FAR


      MOV AX,4C00H       ;SALIDA AL DOS
      INT 21H

EXIT ENDP

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

CAMBIO PROC FAR
call getpagina ; se llama la pagina actual:P
;PROCESO PARA CAMBIAR DE LINEA DESPUES DE QUE YA TERMINO DE ESCRIBIR
ADD CONTADOR,01H
MOV AL,CONTADOR
MOV AH, 02H       ; PETICIÓN
MOV BH, pagina          ; NÚMERO DE PÁGINA
MOV DH, CONTADOR        ; FILA x
MOV DL, 0         ; COLUMNA 0 para empezar la linea
INT 10H
mov aux,0h

RET

CAMBIO ENDP
```

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

INSERT PROC FAR

; ESTO HACE LA PETICION PARA ESCRIBIR SUPUESTAMENTE 40 CARACTERES POR LINEA
; PARA CON ENTER Y LLENA DE ESPACIOS
mov ah,01h

ingreso:
int 21h
inc aux
cmp ax,00 ; es tecla de funcion ?
je clinea
CMP   Ax, 0109h         ; ¿ES TAB???
JE    TABs        ; SI
cmp ax, 010dh  ; es enter ??
je clinea ; si si
cmp ax, 0100h ; es home ?
je chome ; si es home lo q presiono
cmp ax, 0100h ; es end ?
je cend
cmp aux, 40h
ja clinea
jmp ingreso



clinea:

CALL CAMBIO

ret

COPIS: CALL COPY
RET
TABs: ;call backspace
CALL TAB
RET
cend: call doend
chome: call home
ret
INSERT ENDP

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

START PROC FAR

MOV AH, 05H       ; PETICIÓN
MOV AL, 00H ; NÚMERO DE PÁGINA
INT 10H

; PARA HACER UN CUADRITO DE COLOR PARA METER LAS INSTRUCCIONES ADENTRO :)

MOV AX, 0605H           ; AL 05, 5 LINEAS
MOV BH, 61H       ; FONDO CAFÉ CON PRIMER PLANO AZUL
MOV CX, 0000H           ; DESDE FILA 10, COLUMNA 28
MOV DX, 0fFFFH          ; HASTA FILA 14, COLUMNA 52
INT 10H

; PETICION PARA POSICIONAR EL CURSOR :P
```

```
MOV AH, 02H       ; PETICIÓN
MOV BH, 0         ; NÚMERO DE PÁGINA EMPIEZA EN LA 0
MOV DH, 00H       ; FILA 0
MOV DL, 0H        ; COLUMNA 0 (MOV DX, 050CH)
INT 10H

CICLO:

      CMP CONTADOR,09H
      JE EXIT1
      CALL INSERT

JMP CICLO

RET

EXIT1: CALL EXIT
START ENDP

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

BEGIN PROC FAR


        MOV AX, @DATA   ; INICIALIZAR AREA DE DATOS
            MOV DS, AX
        MOV AH, 00H     ; CARGO A AH 00H
        MOV AL, 03H     ; ASIGNA MODO TEXTO A COLOR 80X25
        INT 10H

      CALL START  ;LLAMA AL PROCEDIMIENTO Q EMPIEZA A DIBUJAR

      MOV AX,4C00H       ;SALIDA AL DOS
      INT 21H

BEGIN   ENDP
      END BEGIN


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

```
; Nery Guzman
;2007 Universidad del Valle de Guatemala
;http://docs.huihoo.com/help-pc/ pagina de referencia donde estan las
interrupciones


; programa editor de texto mas avanzado que permite crear archivos en una
locacion
; y abrir archivos de texto en distintas paginas de memoria


title
; todo list
; copy paste
;find replace
ENTERS MACRO ESTO
      MOV AH, 02H
      MOV DL, ESTO
      INT 21H
ENDM
.model small
.stack 64
.data
;---------------------------------------------------------------------
; definicion de datos
arreglo                 db          80 dup (?), '$'   ;área de caracteres a
copiar
entrada                 db          80 dup (?), '$'   ; área de entrada
copi              db    80 dup ()
dataToWrite2      db          '', 0dh, 0ah, '', '$'
contador          db          0
cont              db          0
tabulador         db          '    ','$'
fila              db          0
aux               db          0
loc               db          0
renglon           db          0
pagina                  db          0
pag                     db          0
aski              db    0
cursor                  db          0
kursor                  dw          0
handler                 dw          ? ;variable to store file handle
filename          db          "a:\test.txt",0 ;file to be opened
filename2         db          "a:\manual.txt",0 ;file to be opened
filename1         db          "a:\test2.txt",0 ;file to be opened
filename4         db          "a:\test3.txt",0 ;file to be opened
filename3         db          "a:\test4.txt",0 ;file to be opened

filedescriptor    dw          ?
datatowrite       db          80 dup (?)   , 0dh, 0ah,  '$'
errorcreating     db          'error al crear el archivo', 0dh, 0ah, '$'
errorwriting      db          'error al escribir en el archivo', 0dh, 0ah,'$'
success                 db          'archivo creado con exito', 0dh, 0ah, '$'
pressakey         db          'presione una tecla para salir $'
blur              dw          0
openerror         db          "an error has occured(opening)!$"
readerror         db          "an error has occured(reading)!$"
buffer            db          101 dup (?) ;buffer to store data from
zum                     db          0
zum2              db          0
bye                     db          '                       Desea guardar el
archivo', '$'
bye2              db          '                               Y/N             ',
```

```
'$'
guardar           db          '                          Su archivo ha sido
guardado', '$'
nosave            db      '                          No se guardo el archivo',
'$'
cargar            db      'Su archivo ha cargado exitosamente','$'

                                  ;file one bigger for $
.code
;-------------------------------------------------------------------------
getpagina proc far
      mov ah,0fh
      int 10h                  ; se guarda en bh el numero de pagina actual
      mov pagina, bh
      ret
getpagina endp
;-------------------------------------------------------------------------
grabararchivo proc far
call createfile
      jc cerror
      call writefile
      jc werror
      lea dx, success
      call cout
      jmp endprogram
cerror:
      lea dx, errorcreating
      call cout
      jmp endprogram
werror:
      lea dx, errorwriting
      call cout
endprogram:
      lea dx, pressakey
      call cout
      call cin
      mov ax, 4c00h                ; salir al dos
      int 21h
      ret
grabararchivo endp
;-----------------------------------------------------------------
createfile proc near
      mov ax, 3c00h                      ; crear archivo
      xor cx, cx                         ; cx = 0, atributo normal
      lea dx, filename             ; nombre del archivo
      int 21h
      mov filedescriptor, ax       ;guarda el handler
      ret
createfile endp                    ;guardar el manejador
;-----------------------------------------------------------------
writefile proc near
      mov pagina, 0
      call home
      call getcursor
      mov dx, 0000h
lup:
      ;call backspace
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página
      mov dh, zum       ; columna 0 (mov dx, 050ch)
      mov dl, zum2
      int 10h
      mov blur, 0000h
      mov bx , 0000h
```

```
copiar2:
      mov bh,00h              ;se usa para poner el cursor al principio
    mov ah, 08h
      mov bh, pagina
    int 10h
      mov bx, blur
    mov entrada[bx], al  ; de el registro se pasa al buffer de copiado
    call mright      ; se avanza hacia la derecha
      inc blur
      cmp bl, 80        ; compara con el limite de caracteres
      jb copiar2       ; si es menor brina a hacer el ciclo de nuevo
      mov ax, 4000h                       ;escribir a archivo
      mov bx, filedescriptor       ;carga el nombre del archivo
      mov cx, 80                          ; numero de bytes a escribir
      lea dx, entrada              ;data
      int 21h
      mov ax, 4000h                       ;escribir a archivo
      mov bx, filedescriptor
      mov cx, 2
      lea dx, datatowrite2         ;se usa para ingresar o simular los enter
depues de 80 caracteres :p
      int 21h
      inc zum    ; se incrementa el contador
      cmp zum, 21 ; se compara con las 14 lineas posibles en el editor
      jne lup
      inc pagina
      cmp pagina, 3
      jb lup
      ret
writefile endp
;---------------------------------------------------------------
cout proc near
      mov ax, 0900h
      int 21h
      ret
cout endp
;---------------------------------------------------------------
cin proc near
      mov ax, 0a00h
      int 21h
      ret
cin endp
;---------------------------------------------------------------
getcursor proc far
      call getpagina     ;este proceso sirve para conseguir las coordenadas del
cursor en cualquier momento
      mov ah,03h
      mov bh,pagina
      int 10h
      mov ax,dx
      mov cursor,dh
      ret
getcursor endp
;-------------------------------------------------------------------------
ponerhandler proc far
cmp pagina,0
je paginaz
cmp pagina,1
je pagina1
cmp pagina,2
je pagina2
cmp pagina,3
je pagina3
```

```
paginaz:
      mov pagina,0
      lea dx,FileName  ;put address of fileneame in dx
      mov al,2         ;access mode - read and write
      mov ah,3Dh       ;function 3Dh -open a file
      int 21h          ;call DOS service
      mov Handler,ax    ;save file handle for later
      ;cmp Handler, 0
      ;jc Errorabrir2a
      mov bp,3
      ret
pagina1:
      mov pagina,1
      lea dx,FileName  ;put address of fileneame in dx
      mov al,2         ;access mode - read and write
      mov ah,3Dh       ;function 3Dh -open a file
      int 21h          ;call DOS service
      mov Handler,ax    ;save file handle for later
      ;cmp Handler, 0
      ;jc Errorabrir2a
      mov bp,3
      ret
pagina2:
      mov pagina,2
      lea dx,FileName  ;put address of fileneame in dx
      mov al,2         ;access mode - read and write
      mov ah,3Dh       ;function 3Dh -open a file
      int 21h          ;call DOS service
      mov Handler,ax    ;save file handle for later
      ;cmp Handler, 0
      ;jc Errorabrir2a
      mov bp,3
      ret
pagina3:
      mov pagina,3
      lea dx,FileName  ;put address of fileneame in dx
      mov al,2         ;access mode - read and write
      mov ah,3Dh       ;function 3Dh -open a file
      int 21h          ;call DOS service
      mov Handler,ax    ;save file handle for later
      ;cmp Handler, 0
      ;jc Errorabrir2a
      mov bp,3
      ret
ponerhandler endp
;--------------------------------------------------------------------------
abrirarchivo proc far

lea dx,filename  ;nombre del archivo
lupi22:
      mov al,2         ;funcion de leer y escribir
      mov ah,3dh       ;funcion para abrirlo
      int 21h
      mov handler,ax    ;grabar el manejador
      cmp handler, 0
      jc errorAbrir22
      mov bp, 10

salto_x22:
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página
      mov dl, 0h        ; columna 0 (mov dx, 050ch)
      int 10h
```

```
      mov dx,offset buffer  ;se direcciona el buffer en dx
      mov bx,handler
      mov cx,90         ;los bytes q se leeran del archivo
      mov ah,3fh            ;funcion para leer
      int 21h
      cmp handler, 0
      jc errorLeer22
      mov di,offset buffer+101  ;sirve para decir donde poner el terminador de
cadena
      mov byte ptr [di],"$"     ;lo pone en [es:di]
      mov ah,9                  ;imprime
      mov dx,offset buffer
      int 21h
      dec bp
      cmp bp, 0
      jne salto_x22
      mov bx,handler
      mov ah,3eh
      int 21h
      ret
errorAbrir22:
      mov dx,offset openerror
      mov ah,09h
      int 21h
     mov ax,4c01h
      int 21h
ret
errorLeer22:
      mov dx,offset readerror
      mov ah,09h
      int 21h
      mov ax,4c02h
        int 21h
            ret


    mov ah, 09h
    lea dx, cargar
    int 21h
      ret



abrirarchivo endp


;-------------------------------------------------------------------------
home proc far
      call getpagina
      call getcursor
      mov kursor,dx
      ;call backspace
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página
      mov dl, 0h        ; columna 0 (mov dx, 050ch)
      int 10h
      mov aux,0h
      mov contador,0h
      ret
home endp
;-------------------------------------------------------------------------
tab proc far
      call getcursor
      cmp dl, 55
```

```
      ja nohacer               ; para que no deje hacer tabulador fuera del
limite de caracteres :p
      add dl,04h
      mov ah, 02h       ;      petición
      mov bh, pagina          ; que hace que corra el cursor 4 espacios para
tabular
      int 10h
      add aux,04h             ; se le restan 4 caracteres al texto actual
nohacer:
      ret
tab endp
;-------------------------------------------------------------------------copiar
proc far

exit proc far
      mov ax,4c00h       ;salida al dos
      int 21h
exit endp
;---------------------------------------------------------------------------
goend proc far
      call getpagina
      call getcursor
      mov dx, kursor
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página
      int 10h
      mov contador,0h
      ret
goend endp
;---------------------------------------------------------------------------
cambio proc far
      call getpagina           ; se llama la pagina actual
                                    ;proceso para cambiar de linea despues de
que ya termino de escribir
      add contador,01h
      mov al,contador
      mov ah, 02h             ; petición
      mov bh, pagina          ; número de página
      mov dh, contador        ; fila x
      mov dl, 0               ; columna 0 para empezar la linea
      int 10h
      mov aux,0h
      ret
cambio endp
;---------------------------------------------------------------------------
ascii proc far
      call getpagina    ;se llama las coordenadas de la pagina
      mov ah,0ah              ; se llama al modo ingresar caracter
      mov al,aski       ; se guarda el aski metido anteriormente en la variable
aski
      mov bh,pagina     ;se pone la pagina actual
      mov cx,01               ;se dice cuantas veces se va a imprimir en la
pocision del cursor esa letra
      int 10h                 ; zaz interrupcion
      inc aux                 ; se incrementa la linea
      call getcursor    ; se llaman las coordenadas del cursor para avanzar
      add dl,01h              ; se avanza un espacio el cursor
      mov ah, 02h       ; petición
      mov bh, pagina    ; se pone la pagina
      int 10h                 ; zaz avanza uno el cursor para simular el avanze
de un space despues de un tecladazo :0)
ret
ascii endp
;---------------------------------------------------------------------------
```

```
insert proc far
      ; esto hace la peticion para escribir 80 caracteres por linea
      ; para con enter y llena de espacios
ingreso:
      mov ah,10h
      int 16h
      mov aski, al
      cmp al,00h        ; es tecla de funcion ?
      je cfuncion             ; si si brinka a teclado extendido
      cmp al,0e0h             ;es tecla de teclado extendido?
      je ctcladoex             ; si si brinka a teclado extendido
      cmp   ax, 0f09h   ; ¿es tab???
      je    tabs        ; si
      cmp ax, 0e08h           ;es backspace?
      je cback                ;si si brinka a backspace :p
      cmp ax, 1c0dh     ; es enter ??
      je clinea          ; si si
      cmp ax, 2e03h ;es ctrl c
      je copis
      cmp ax, 2f16h
      je paste
      cmp ax,  2d18h
      je cut
      call ascii        ; si no  es tecla especial brinka a meter el asciii
      cmp aux, 40h            ;compara si ya llego al final de la linea
adimitido
      ja clinea         ;si ya llego entonces brinka a cambio de linea simula
enter
jmp ingreso                   ;si no simula de nuevo la entrada para seguir
metiendo askis
clinea:
      call cambio
      ret
copis:
      call copiar
      ret
paste:
      call pegar
      ret
cut:
      call cortar
      ret
cback:
      call getpagina
      call backspace
      ret
tabs:
      call getpagina
      call tab
      ret
cfuncion:
      mov aux,0h
      mov contador,0h
      call funcionloka
      ret
ctcladoex:
      call tecladoextra
      ret
insert endp
;-------------------------------------------------------------------------
copiar proc far
      call getpagina
      call home
      mov blur, 0000h
```

```
      mov bx , 0000h
copyloko:
      copying:
      mov bh,00h
    mov ah, 08h
      mov bh, pagina
    int 10h
      mov bx, blur
    mov arreglo[bx], al  ; de el registro se pasa al buffer de copiado
    call mright      ; se avanza hacia la derecha
      inc blur
      cmp bl, 80
      jb copying
      ret
copiar endp
;--------------------------------------------------------------------------
pegar proc near
      call home
      mov ah, 09h       ;muestra mensaje en pantalla
      lea dx, arreglo
      int 21h
ret
pegar endp
;--------------------------------------------------------------------------
cortar proc near
mov bx, 0
      mov cx, 10
      call home
dcut:
      mov al, [es:di]
      mov arreglo[bx], al
      mov [es:di], 720h
      inc bx
      push cx
      call mright
      pop cx
      loop dcut
      ret
cortar endp
;--------------------------------------------------------------------------
;--------------------------------------------------------------------------
backspace proc far
      call getcursor
      cmp dl,0h
      je dont
      add dl,-01h             ; se atraza un espacio el cursor
      mov ah, 02h       ; petición
      mov bh, pagina    ; se pone la pagina
      int 10h                 ; zaz avanza uno el cursor para simular el avanze
de un space despues de un tecladazo :0)
      mov aski,20h
      call ascii
      add dl,-01h             ; se atraza un espacio el cursor
      mov ah, 02h       ; petición
      mov bh, pagina    ; se pone la pagina
      int 10h                 ; zaz se atrasa 2 veces porque el metodo ascii
avanza uno entonces solo hubiera dejado un espacio en blanco dx !
dont:
      ret
backspace endp
;--------------------------------------------------------------------------
tecladoextra proc far
      cmp ax, 47e0h           ;es home ?
      je chome                ;si es home lo q presiono
```

```
      cmp ax, 4fe0h          ;es end?
      je cend                      ;si es end brinka
      cmp ax, 49e0h          ;es page up?
      je cpageup             ;si si entonces brikca
      cmp ax, 51e0h          ;es page down?
      je cpagedown           ;si es igual entonces brinca a con pagedown
      cmp ax, 4be0h          ;es flechita a la izquierda?
      je cizkierda
      cmp ax, 4de0h          ;es flechita a la derecha?
      je cderecha
      cmp ax,50e0h                          ; es flecha abajo
      je cdown
      cmp ax, 48e0h          ;es flecha arriba
      je cup
cup:
      call movup
      ret
cdown:
      call movdown
      ret
cderecha:
      call mright
      ret
cizkierda:
      call mleft
      ret
cpagedown:
      call pages
      ret
cpageup:
      call pages
      ret
chome:
      call home
      ret
cend:
      call goend
      ret
tecladoextra endp
;-------------------------------------------------------------------------
funcionloka proc far
cmp ah, 3bh
            jne s1
            call f1
      s1:
            cmp ah, 3ch
            jne s2
            call f2
      s2:
            cmp ah, 3dh
            jne s3
            call f3
      s3:
            cmp ah, 3eh
            jne s4
            call f4
      s4:
            cmp ah, 3fh
            jne s5
            call f5
      s5:
            cmp ah, 41h
            je goout
            cmp ah, 01h
```

```
            je goout
            cmp ah, 49h
            jne s7
            call pages
      s7:
            cmp ah, 51h
            jne s8
            call pages
      s8:
            cmp ah, 53h
            jne s9
            call delete
      s9:
            cmp ax, 2e03h
            jne s10
            call copiar
            jmp goout
      s10:
            cmp ax, 2e10h
            jne s11
            call pegar
      s11:
            cmp ax, 2e18h
            jne s12
            call cortar
      s12:
            jmp final
goout:
call salida
final:
ret
funcionloka endp
;-------------------------------------------------------------------------
menu proc far
cuatro:
      mov ah, 05h       ; petición
      mov al, pagina    ; número de página
      int 10h
      ; hace un cuadro de color para meter las instrucciones adentro
      mov ax, 0605h           ; al 05, 5 lineas
      mov bh, 61h             ; fondo café con primer plano azul
      mov cx, 0000h           ; desde fila 10, columna 28
      mov dx, 0ffffh          ; hasta fila 14, columna 52
      int 10h
      add pagina,1
      cmp pagina, 4
      jb cuatro
ret
menu endp
;-------------------------------------------------------------------------
delete proc near
      mov [es:di], 720h
      ret
delete endp
;-------------------------------------------------------------------------
start proc far
      mov ah, 05h       ; petición
      mov al, 00h ; número de página
      int 10h
      mov ah,02h
      mov bh, 00h
      mov dx, 0000h
      ; hace un cuadro de color para meter las instrucciones adentro
      call menu
```

```
ciclo:
      cmp contador,14h
      je exit1
      call insert
      jmp ciclo
      ret
exit1:
      call exit
start endp
;-------------------------------------------------------------------------
;cuando se presiona f1, f2, f3 o f4 estos procesos se encargan de
;cambiar de pagina de la 0 a la 3 y colocar el cursor en la fila
;0 y la columna 0
;-------------------------------------------------------------------------
f1 proc near
      mov pagina, 0
      call abrirarchivo

      ret
f1 endp
;-------------------------------------------------------------------------
f2 proc near
      call grabarArchivo
      ret
f2  endp
;-------------------------------------------------------------------------
f3 proc near
      mov ah, 05h
      mov al, 2
      int 10h
      mov ah, 02h
      mov bh, 2
      mov dh, 05
      mov dl, 12
      int 10h
      call getpagina
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página empieza en la 0
      mov dh, 00h       ; fila 0
      mov dl, 0h        ; columna 0 (mov dx, 050ch)
      int 10h
      mov aux, 00
      ret
f3  endp
;-------------------------------------------------------------------------
f4 proc near


      ret

f4 endp

;-------------------------------------------------------------------------
;Este proceso se encarga de limpiar lo que se encuentra en la pagina
;actual
;-------------------------------------------------------------------------
f5 proc near
lea dx,filename2  ;nombre del archivo
lupi2:
      mov al,2         ;funcion de leer y escribir
      mov ah,3dh       ;funcion para abrirlo
      int 21h
      mov handler,ax    ;grabar el manejador
```

```
      cmp handler, 0
      jc errorAbrir2
      mov bp, 10

salto_x2:
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página
      mov dl, 0h        ; columna 0 (mov dx, 050ch)
      int 10h

      mov dx,offset buffer  ;se direcciona el buffer en dx
      mov bx,handler
      mov cx,90         ;los bytes q se leeran del archivo
      mov ah,3fh            ;funcion para leer
      int 21h
      cmp handler, 0
      jc errorLeer2
      mov di,offset buffer+101  ;sirve para decir donde poner el terminador de
cadena
      mov byte ptr [di],"$"     ;lo pone en [es:di]
      mov ah,9                  ;imprime
      mov dx,offset buffer
      int 21h
      dec bp
      cmp bp, 0
      jne salto_x2
      mov bx,handler
      mov ah,3eh
      int 21h
      ret
errorAbrir2:
      mov dx,offset openerror
      mov ah,09h
      int 21h
     mov ax,4c01h
      int 21h
ret
errorLeer2:
      mov dx,offset readerror
      mov ah,09h
      int 21h
      mov ax,4c02h
        int 21h
            ret


    mov ah, 09h
    lea dx, cargar
    int 21h
      ret

f5 endp
;-------------------------------------------------------------------------
;cuando se utilizan las teclas pageup y pagedown invocan a este metodo
;para poder recorrer de la pag 0 a la 3 de forma circular
;-------------------------------------------------------------------------
pages proc near
    cmp pag, 3
    jne change
    mov pag, -1
change:
      inc pag
      mov ah, 05h    ; petición
      mov al, pag    ; número de página
```

```
      int 10h
      mov ah, 02h
      mov bh, 3
      mov dh, 00     ; esta parte se encarga de colocar el cursor
      mov dl, 00     ; en la fila y columna 0.
      int 10h
    ret
pages endp
;-------------------------------------------------------------------------
;este proceso se encarga del movimiento de las flechas izquierda y
;derecha
;-------------------------------------------------------------------------
mleft proc far
      call getcursor
      mov kursor, dx
      cmp dl,0h
      je fin5
      mov ah, 02h       ; petición
      add dl,-01h
      mov bh, pagina    ; número de página empieza en la 0
      int 10h
      add aux, -01h
fin5:
ret
endp
;-------------------------------------------------------------------------
mright proc far
      call getcursor
      mov kursor, dx
      cmp dl,40h
      je fin4
      mov ah, 02h       ; petición
      add dl,01h
      mov bh, pagina    ; número de página empieza en la 0
      int 10h
      add aux, 01h
fin4:
ret
endp
;-------------------------------------------------------------------------
movdown proc near
      call getcursor
      mov kursor, dx
      cmp dh,13h
      je fin3
      mov ah, 02h       ; petición
      add dh,1h
      mov bh, pagina    ; número de página empieza en la 0
      int 10h
fin3:
ret
movdown endp
;-------------------------------------------------------------------------
movup proc near
      call getcursor
      mov kursor, dx
      cmp dh,0
      je fin2
      mov ah, 02h       ; petición
      add dh,-1h
      mov bh, pagina    ; número de página empieza en la 0
      int 10h
fin2:
ret
```

```
movup endp

;-------------------------------------------------------------------------
begin proc far
      mov ax, @data   ; inicializar area de datos
      mov ds, ax
      mov ah, 00h     ; cargo a ah 00h
      mov al, 03h     ; asigna modo texto a color 80x25
      int 10h
      call start  ;llama al procedimiento q empieza a dibujar
salida:
      mov ah, 05h ; salida de protocolo dx
      mov al, 0
      int 10h
      mov ax, 0600h
      mov bh, 07h
      mov cx, 0000
      mov dx, 184fh
      int 10h
      mov ax,4c00h       ;salida al dos
      int 21h
begin   endp
end begin
;-------------------------------------------------------------------------
```

```
;*****************************************************************
; UNIVERSIDAD DEL VALLE DE GUATEMALA
; FACULTAD DE INGENIERIA
;     NERY GUZMAN
;     WALTER SEGURA 06151
; PROGRAMACION EN ASSEMBLER
; CATEDRATICA: MARTHA LIGIA NARANJO
; PROGRAMA EL CUAL REALIZA CUATRO PORCESOS SIMULTANEOS
;*****************************************************************

;                          PROYECTO 3

METER MACRO             ;MACRO PARA GUARDAR LOS REGISTROS ACTUALES
     PUSH SI
     PUSH CX
       PUSH AX
       PUSH BX
       PUSH DX
       MOV AX, 0
       MOV AL, COLUMNA
       PUSH AX
       MOV AL, FILA
       PUSH AX
ENDM
;-------------------------------------------------------------------------
SACAR MACRO                  ;MACRO PARA REESTABLECER LOS REGISTROS
    POP AX
      MOV FILA, AL
      POP AX
      MOV COLUMNA, AL
    POP DX
      POP BX
      POP AX
    POP CX
    POP SI
ENDM
;-------------------------------------------------------------------------------
INIC MACRO NUMI, X                ;MACRO DE INICIALIZACION
LOCAL CICLOINIC1             ;RECIBE DE PARAMETRO EL NUMERO DE PROCESO
LOCAL CICLOINIC2
LOCAL CICLOINIC3
LOCAL FINALIC

      MOV BP, WORD PTR VECTORBP[NUMI]   ;MUEVE AL BS Y AL SP LOS VECTORES
CORRESPONDIENTES
    MOV SP, WORD PTR VECTORSP[NUMI]
    PUSHF
    MOV AL, X
    CMP AL, 2
    JE CICLOINIC1
    CMP AL, 3
    JE CICLOINIC2
    CMP AL, 4
    JE CICLOINIC3

    CICLOINIC1:
    PUSH SEG CUADRANTE2
      PUSH OFFSET CUADRANTE2
      METER
      MOV [WORD PTR VECTORSP+NUMI], SP
      JMP FINALIC
```

```
      CICLOINIC2:
    PUSH SEG CUADRANTE3
      PUSH OFFSET CUADRANTE3
      METER
      MOV [WORD PTR VECTORSP+NUMI], SP
      JMP FINALIC

      CICLOINIC3:
    PUSH SEG CUADRANTE4
      PUSH OFFSET CUADRANTE4
      METER
      MOV [WORD PTR VECTORSP+NUMI], SP

FINALIC:

ENDM
;-------------------------------------------------------------------------
.MODEL LARGE
    VSEGMENT    SEGMENT    AT 0B800H
.STACK 64
; ---------------------------------------------------------------
.DATA

ATRIBUTO DB  10H      ;VARIABLE DEL ATRIBUTO DE LA PRIMERA PAGINA
FILA    DB  2
COLUMNA   DB  0
CODIGO      DB    ?          ;VARIABLE QUE GUARDA LA OPCION
VECTORBP          DW    4 DUP(1000H)
VECTORSP          DW    4 DUP(1000H)
RELOJVIEJO        DD    ?           ; GUARDA DIRECCIONES ANTIGUAS
TECLADOVIEJO      DD    ?
OPCION           DW 0
BANDERA                DB    ?           ; BANDERA PARA TERMINAR
NUMI            DB  0
X               DB  0

;-----------------------------------------------------------------
; INICIO DEL CODIGO
.CODE
;-----------------------------------------------------------------
; PROCEDIMIENTO PARA EL DESPLIEGUE DE LA PAGINA 1
UNO   PROC NEAR

          MOV AH, 05H         ; PETICIÓN
          MOV AL, 0     ; NÚMERO DE PÁGINA
            INT 10H
          MOV AX,0B800H ; DIRECCIÓN DE INICIO DE MEMORIA DE VIDEO PAGINA 0
        MOV ES,AX         ; SE CARGA AL REGISTRO ES LA DIRECC MEM
          MOV DI,0

      RET

UNO   ENDP
;-----------------------------------------------------------------------
;PROCEDIMIENTO QUE CREA EL DIBUJO DE LOS CUADRANTES 1 Y 4
DIBUJO1 PROC NEAR
CMP ATRIBUTO, 57H
JE  RESETAT
SUMAAT:
ADD ATRIBUTO, 1H

CALL MOSTRAR
CALL DIBUJAR1
JMP DIBUJOS1RET
```

```
RESETAT:
MOV ATRIBUTO, 10H
JMP SUMAAT

DIBUJOS1RET:
RET
DIBUJO1 ENDP
;-------------------------------------------------------------------------
;PROCEDIMIENTO QUE CREA EL DIBUJO DE LOS CUADRANTES 2 Y 3
DIBUJO2  PROC NEAR
CMP ATRIBUTO, 37H
JE  RESETAT2
SUMAAT2:
ADD ATRIBUTO, 1H
CALL MOSTRAR
CALL DIBUJAR
CALL DIBUJAR
CALL DIBUJAR
CALL DIBUJAR
JMP DIBUJOS2RET

RESETAT2:
MOV ATRIBUTO, 10H
JMP SUMAAT2

DIBUJOS2RET:
RET
DIBUJO2 ENDP
;-------------------------------------------------------------------------
;PROCEDIEMINTEO EL CUAL DIBUJA UN CUADRADO CUANDO ES LLAMADO
DIBUJAR1 PROC NEAR
CALL DERECHA
CALL DERECHA
CALL DERECHA
CALL DERECHA
CALL DERECHA

CALL VERTICALUP
INC COLUMNA
INC COLUMNA
CALL MOVE
CALL IZQUIERDA
CALL IZQUIERDA
CALL IZQUIERDA
CALL IZQUIERDA
CALL IZQUIERDA
DEC COLUMNA
CALL MOVE
CALL MOSTRAR

CALL VERTICALDOWN

DIBUJO1RET:
RET
DIBUJAR1 ENDP
;-------------------------------------------------------------------------
;PROCEDIMENTO QUE DIBUJA UNA PARTE DEL DIBUJO 2
DIBUJAR PROC NEAR
CALL DERECHA
CALL VERTICALDOWN
CALL DERECHA
CALL VERTICALUP
```

```
RET
DIBUJAR ENDP
;-------------------------------------------------------------------------
;SE MUEVE CINCO ESPACIOS A LA DERECHA Y EN CADA ESPACIO MUESTRA UN CARACTER
DERECHA PROC NEAR

INC COLUMNA
CALL MOVE
CALL MOSTRAR

INC COLUMNA
CALL MOVE
CALL MOSTRAR

INC COLUMNA
CALL MOVE
CALL MOSTRAR

INC COLUMNA
CALL MOVE
CALL MOSTRAR

INC COLUMNA
CALL MOVE
CALL MOSTRAR

RET
DERECHA ENDP
;-------------------------------------------------------------------------
;SE MUEVE 5 ESPACIOS  A LA IZQUIERDA IMPRIMIENDO EL CARACTER EN CADA ESPACIO
IZQUIERDA PROC NEAR

DEC COLUMNA
CALL MOVE
CALL MOSTRAR

DEC COLUMNA
CALL MOVE
CALL MOSTRAR

DEC COLUMNA
CALL MOVE
CALL MOSTRAR

DEC COLUMNA
CALL MOVE
CALL MOSTRAR

DEC COLUMNA
CALL MOVE
CALL MOSTRAR

IZRET:
RET
IZQUIERDA ENDP
;-------------------------------------------------------------------------
;PROCEDIENTO EL CUAL EL SE MUEVE HACIA ARRIBA IMPRIMIENDO EL CARACTER
VERTICALUP PROC NEAR

INC FILA
INC FILA
CALL MOVE
CALL MOSTRAR
INC FILA
```

```
INC FILA
CALL MOVE
CALL MOSTRAR
INC FILA
INC FILA
DEC COLUMNA
CALL MOVE

RET
VERTICALUP ENDP
;-------------------------------------------------------------------------
;PROCEDIENTO EL CUAL EL SE MUEVE HACIA ABAJO IMPRIMIENDO EL CARACTER
VERTICALDOWN PROC NEAR

DEC FILA
DEC FILA
CALL MOVE
CALL MOSTRAR
DEC FILA
DEC FILA
CALL MOVE
CALL MOSTRAR
DEC FILA
DEC FILA
DEC COLUMNA
CALL MOVE

RET
VERTICALDOWN ENDP
;-------------------------------------------------------------------------
;POSICIONA EL CURSOR DEPENDIENDO LAS VARIABLES FILA Y COLUMNA
MOVE PROC NEAR
      MOV AH, 02H                       ; SE PIDE UN SET DE CURSOR
      MOV BH, 0                    ; EN LA PAGINA EN DONDE SE DESEA
      MOV DL, COLUMNA                   ; EN LA COLUMNA QUE SE QUIERE
      MOV DH, FILA                      ; EN LA FILA QUE SE QUIERE
      INT 10H

      RET
MOVE ENDP
;-------------------------------------------------------------------
;MUESTRA UN CARACTER EN DONDE SE ENCUENTRE EL CURSOR
MOSTRAR PROC NEAR
MOV AH, 09H
MOV AL, 2AH
MOV BL, ATRIBUTO
MOV CX, 1
INT 10H

RET
MOSTRAR ENDP
;-------------------------------------------------------------------
;LIMPIA LA PANTALLA LLENANDOLA DE ESPACIO EN BLANCO
CLEAR PROC NEAR
      MOV AX, 0600H                          ; SE MANDA LA PANTALLA COMPLETA
      MOV BH, 0007H
      MOV CX, 0000H                          ; DESDE LA POSICION 00
      MOV DX, 184FH                          ; HASTA LA ULTIMA POSICION,
ESQUINA INFERIOR DERECHA
      INT 10H
      MOV DI,0                               ; DESDE LA POSICION OO
      MOV CX,2000                            ; SE REPITE DOS MIL VECES QUE ES
PANTALLA COMPLETA
      MOV AX,20H                        ; CARACTER VACIO
```

```
      REP STOSW                                  ; SE IMPRIME EN PANTALLA

      RET
CLEAR ENDP
;-------------------------------------------------------------------------
CUADRANTE1 PROC NEAR
CALL CLEAR
CMP BANDERA, 2
      JE CUADRANTE1RET
CUAD1:
         MOV COLUMNA, 4
         MOV FILA, 4
         CALL MOVE
         CALL DIBUJO1

        MOV AH, 01H        ; SE MIRA EL ESTADO DEL BUFFER
        INT 16H
        JNZ PRIMERO         ; SI SE PRESIONA UNA TECLA DURANTE LA IMPRESION SE
DETIENE
        JMP CUAD1        ;DE LO CONTRARIO SIGUE IMPRIMIENDO

PRIMERO: MOV AH, 07H
        INT 21H
        CMP AL, 31H        ;SI ES LA TECLA 1 SE SALE DEL PROGRAMA
        JE  CUADRANTE1RET
        JMP CUAD1
                ;SI ES CUALQUIER OTRA TECLA SE IGNORA
CUADRANTE1RET:

RET

CUADRANTE1 ENDP
;-----------------------------------------------------------------
CUADRANTE2 PROC NEAR
CMP BANDERA, 2
      JE CUADRANTE2RET
      CUAD2:
              MOV COLUMNA, 1
              MOV FILA,   22
              CALL MOVE
              CALL DIBUJO2


          MOV AH, 01H        ; SE MIRA EL ESTADO DEL BUFFER
        INT 16H
        JNZ SEGUNDO         ; SI SE PRESIONA UNA TECLA DURANTE LA IMPRESION SE
DETIENE
        JMP CUAD2        ;DE LO CONTRARIO SIGUE IMPRIMIENDO

SEGUNDO: MOV AH, 07H
        INT 21H
            CMP AL, 31H        ;SI ES TECLA 1 SE SALE DEL PROGRAMA
        JE CUADRANTE2RET
        JMP CUAD2          ;SI ES CUALQUIER OTRA TECLA SE IGNORA

CUADRANTE2RET:

RET
CUADRANTE2 ENDP
;-------------------------------------------------------------------
CUADRANTE3 PROC NEAR
CMP BANDERA, 2
      JE CUADRANTE3RET
    CUAD3:
```

```
              MOV COLUMNA, 43
              MOV FILA,   10
              CALL MOVE
              CALL DIBUJO2
              MOV AH, 01H        ; SE MIRA EL ESTADO DEL BUFFER
        INT 16H
        JNZ TERCERO         ; SI SE PRESIONA UNA TECLA DURANTE LA IMPRESION SE
DETIENE
        JMP CUAD3        ;DE LO CONTRARIO SIGUE IMPRIMIENDO

TERCERO: MOV AH, 07H
        INT 21H
            CMP AL, 31H
        JE CUADRANTE3RET
        JMP CUAD3

CUADRANTE3RET:

RET
CUADRANTE3 ENDP
;-------------------------------------------------------------------------
CUADRANTE4 PROC NEAR
CMP BANDERA, 2
      JE CUADRANTE4RET
   CUAD4:
        MOV COLUMNA, 50
        MOV FILA, 15
        CALL MOVE
        CALL DIBUJO1
        MOV AH, 01H        ; SE MIRA EL ESTADO DEL BUFFER
        INT 16H
        JNZ CUARTO         ; SI SE PRESIONA UNA TECLA DURANTE LA IMPRESION SE
DETIENE
        JMP CUAD4        ;DE LO CONTRARIO SIGUE IMPRIMIENDO

CUARTO: MOV AH, 07H
        INT 21H
          CMP AL, 31H
        JE CUADRANTE4RET
        JMP CUAD4
CUADRANTE4RET:
RET
CUADRANTE4 ENDP
;-------------------------------------------------------------------------
INICIALIZACIONES PROC NEAR
   ;ACA SE INICIALIZAN LOS SEGMENTOS A UTILIZAR
    MOV WORD PTR [VECTORBP], BP
    MOV WORD PTR [VECTORSP], SP
    MOV CX, 3
      MOV BX, 0
INICIALIZA:
    MOV AX, VECTORBP[BX]           ;SE HACE UN CICLO LLENANDO ASI 4 SEGMENTOS
    ADD BX, 2
    ADD AX, 1000H                  ;CADA UNO DE 1000H
    MOV VECTORBP[BX], AX
    MOV VECTORSP[BX], AX
    LOOP INICIALIZA
    MOV [WORD PTR VECTORSP], SP
    MOV CX, 3
    INIC 2, 2                           ;POR MEDIO DE ESTE MACRO SE DIRECCIONA
CORRECTAMENTE
    INIC 4, 3                           ;CADA PROCESO CON SU RESPECTIVA AREA
DE DATOS.
    INIC 6, 4                      ; Y CADA CUADRANTE EN SI.
```

```
      MOV SP, [WORD PTR VECTORSP] ;LUEGO SE METE EN SP Y BP LAS COORDENADAS
CORRESPONDIENTES
      MOV BP, [WORD PTR VECTORBP]
    RET
INICIALIZACIONES ENDP
;-------------------------------------------------
ACTUALIZAR PROC NEAR
    CMP OPCION, 6       ;ACA SE COMPARA
    JE GIRAR
    ADD OPCION, 2
    JMP ACTUALIZARET
      GIRAR:
      MOV OPCION, 0
      ACTUALIZARET:
    RET
ACTUALIZAR ENDP
;-------------------------------------------------------------------------------
-------------------------------------------------
RELOJNUEVO PROC NEAR
    CLI                                         ;NO DEJA INTERRRUPCIONES EXTERNAS
    PUSHF
    CALL DWORD PTR RELOJVIEJO    ; SE GUARDA EL RELOJ VIEJO
      METER                                ; SE LLAMA AL MACRO QUE HACE PUSH A
LOS REGS
    MOV BX, OPCION                         ;SE INGRESA EN BX LA OPCION
    MOV [OFFSET VECTORSP+BX], SP ; SE MUEVE EL VALOR DEL SP AL VECTOR ACTUAL
    CALL ACTUALIZAR
    MOV BX, OPCION                  ; CON OPCION SE SABE QUE PROCEDIMIENTO SE
USARA
    MOV SP, [OFFSET VECTORSP+BX] ; SE METE OPCION EN BX Y SE UTILIZA PARA EL
NUEVO SP
    SACAR                           ; SE HACE POP A LOS REGS
    STI
    IRET

RELOJNUEVO ENDP
;-------------------------------------------------------------------------------
-------------------------------------------------
NUEVORELOJ PROC NEAR
    MOV AH, 35H                                 ;INT DEL RELOJ
    MOV AL, 08H
    INT 21H
    MOV [WORD PTR RELOJVIEJO+2], ES ;SE METE LA DIRECCION DEL ES AL RELOJ VIEJO
+2
    MOV [WORD PTR RELOJVIEJO], BX    ;SE METE EL BX A LA DIRECCION DEL
RELOJVIEJO
    PUSH DS
    MOV AX, SEG RELOJNUEVO              ;SE OBTIENE LA DIRECCION DE DONDE ESTA
LA ETIKETA
    MOV DS, AX                                  ;SE PONE EN LA DIRECCION OBTENIDA
    MOV DX, OFFSET RELOJNUEVO     ;GUARDAMOS EL DESPLAZAMIENTO DE RELOJNUEVO
    MOV AH, 25H                                 ;INT DEL RELOJ
    MOV AL, 08H
    INT 21H
    POP DS
    RET
NUEVORELOJ ENDP
;-------------------------------------------------------------------------------
-------------------------------------------------
TECLADONUEVO PROC NEAR
      CLI
      PUSHF
      CALL DWORD PTR TECLADOVIEJO
      MOV BANDERA, 2
```

```
      STI
      IRET
TECLADONUEVO ENDP
;-------------------------------------------------------------------------------
-------------------------------------------------
NEWKEYBOARD PROC NEAR
      STD
      MOV AH, 35H
    MOV AL, 09H
    INT 21H                                 ;INTERRUPCION DE SETEO DE TECLADO
    MOV [WORD PTR TECLADOVIEJO+2], ES     ;SE MUEVE A LA ETIKETA EL ES ACTUAL
    MOV [WORD PTR TECLADOVIEJO], BX
    PUSH DS
    MOV AX, SEG TECLADONUEVO              ;ACA SE CAPTA LA DIRECCION DE LA
ETIKETA
    MOV DS, AX
    MOV DX, OFFSET TECLADONUEVO                ;ACA SE CAPTA EL OFFSET DE LA
ETIKETA
    MOV AH, 25H
    MOV AL, 09H
    INT 21H                                 ;INTERRUPCION DE NEW KEYBOARD
    POP DS                                              ;SE REESTABLECE DS
      CLD
      RET
NEWKEYBOARD ENDP
;-------------------------------------------------------------------------------
;REGRESA LOS VALORES INICIALES DE LAS INTERRUPCIONES MODIFICADAS
RESTORE PROC NEAR
    CLI
    PUSH DS
    MOV AX, [WORD PTR RELOJVIEJO+2]      ;REGRESA LA INTERRUPCION DEL RELOJ
    MOV DS, AX
    MOV DX, [WORD PTR RELOJVIEJO]
    MOV AH, 25H
    MOV AL, 08H
    INT 21H
    MOV AX, [WORD PTR TECLADOVIEJO+2]     ;REGRESA LA INTERRUPCION DEL TECLADO
    MOV DS, AX
    MOV DX, [WORD PTR TECLADOVIEJO]
    MOV AH, 25H
    MOV AL, 09H
      INT 21H
    POP DS
    STI
    RET
RESTORE ENDP
;-------------------------------------------------------------------------------
--
; PROCEDIMIENTO PRINCIPAL
MAIN  PROC FAR

      MOV AX,@DATA            ; INICIALIZA SEGMENTO DE DATOS
      MOV DS,AX
      CALL UNO
      CALL CLEAR                    ;PROCESO QUE LIMPIA
      CALL INICIALIZACIONES   ;INICIALIZA LOS DATOS Y SEGMENTOS
      CALL NUEVORELOJ               ;PONE EL NUEVO RELOJ
      CALL NEWKEYBOARD        ;PONE EL NUEVO TECLADO
      PUSH   VSEGMENT               ;GUARDA LA PILA
    POP    ES                       ;SACA EL ES ACTUAL
    ASSUME ES:VSEGMENT        ;PONE EN ACCION EL SEGMENTO
    CALL CUADRANTE1           ;COMIENZA EL DIBUJO
      CALL CLEAR                    ;LIMPIA PANTALLA
      CALL RESTORE                  ;SE RESTAURAN LAS INTERRUPCIONES
```

```
SALIDA:
      MOV AH, 05H             ; PETICIÓN
      MOV AL, 0               ; NÚMERO DE PÁGINA
      INT 10H
      MOV AH, 4CH             ; CODIGO DE INTERRUPCI¢N 4CH: SALIDA AL DOS
      INT 21H
MAIN  ENDP

END MAIN
```

```
; Nery Guzman
;2007 Universidad del Valle de Guatemala
;http://docs.huihoo.com/help-pc/ pagina de referencia donde estan las
interrupciones

; editor de texto mas avanzado con teclas y funciones especiales.




.model small
.stack 64
;-------------------------------------------------------------------------
.data                                   ;definicion de datos
entrada          db          80 dup ()   ;área de entrada
arreglo          db    80 dup (?)   ;área de caracteres a copiar
contador    db          0
cont        db          0
tabulador   db          '    ','$'
fila        db          0
aux         db          0
loc         db          0
renglon     db          0
pagina           db          0
pag              db          0
aski        db          0
cursor           db          0
kursor           dw          0
cad0  db    'MENU','$'
cad2  db    'Teclas activas :','$'
cad3  db    'F1,F2,F3 ','$'
cad4  db    'HOME INS','$'
cad5  db    'BACKSPACE DEL AVPAG REPAG','$'
;-------------------------------------------------------------------------
.code
;-------------------------------------------------------------------------
;Metodo para obtener la pagina actual en donde se encuentra el usuario
;-------------------------------------------------------------------------
getpagina proc near

      mov ah, 0fh
      int 10h                                ;se guarda en bh el numero de
pagina actual
      mov pagina, bh
      ret

getpagina endp
;-------------------------------------------------------------------------
;Metodo para obtener las coordenadas del cursor
;-------------------------------------------------------------------------
getcursor proc near

      call getpagina                    ;llama a getpagina para posicionarse
en
      mov ah, 03h                       ;pagina actual
      mov bh, pagina
      int 10h
      mov ax, dx
      mov cursor, dh
      ret

getcursor endp
;-------------------------------------------------------------------------
```

```
;Metodo que nos lleva al principio de la linea en la que nos encontramos
;actualmente
;-------------------------------------------------------------------------
home proc near

      call getpagina
      call getcursor
      mov kursor,dx
      ;call backspace
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página
      mov dl, 0h        ; columna 0 (mov dx, 050ch)
      int 10h
      mov aux,0h
      mov contador,0h
      ret


home endp
;-------------------------------------------------------------------------
;Metodo que se utiliza cuando se presiona la tecla tab entonces deja
;4 espacios
;-------------------------------------------------------------------------
tab proc near

      call getcursor
      mov kursor, dx
      cmp  dl, 55
      ja     terminar         ;esto no permite que el tabulador pase fuera del
limite de caracteres
      add  dl, 04h
      mov  ah, 02h            ;petición
      mov  bh, pagina         ;que hace que corra el cursor 4 espacios para
tabular
      int  10h
      add  aux,04h            ;se le restan 4 caracteres al texto actual

terminar:
      ret

tab endp
;-------------------------------------------------------------------------
;Metodo que se llama con la tecla end esta va al final de la linea en
;donde se encuentra el cursor
;-------------------------------------------------------------------------
fin proc near

      call getpagina
      call getcursor
      mov dx, kursor
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página
      int 10h

      mov contador,0h
      ret

      ret
fin endp
;-------------------------------------------------------------------------
;Metodo que se encarga de cambiar de linea para cuando se esta
;insertando el texto
;-------------------------------------------------------------------------
cambio proc near
```

```
      call getpagina          ;se llama la pagina actual
                                    ;proceso para cambiar de linea despues de
que ya termino de escribir
      add contador,01h
      mov al,contador
      mov ah, 02h             ;petición
      mov bh, pagina          ;número de página
      mov dh, contador  ;fila x
      mov dl, 0               ;columna 0 para empezar la linea
      int 10h
      mov aux,0h
      call getcursor
      mov kursor,dx
      ret

cambio endp
;---------------------------------------------------------------------------
;Metodo para desplegar en pantalla y guardar en la memoria de video
;---------------------------------------------------------------------------
ascii proc near

      call getpagina    ;se llama las coordenadas de la pagina
      mov ah, 0ah     ;se llama al modo ingresar caracter
      mov al, aski      ;se guarda el aski metido anteriormente en la variable
aski
      mov bh, pagina  ;se pone la pagina actual
      mov cx, 01        ;se dice cuantas veces se va a imprimir en la pocision
del cursor esa letra
      int 10h           ;interrupcion
      inc aux           ;se incrementa la linea
      call getcursor    ;se llaman las coordenadas del cursor para avanzar
      add dl, 01h       ;se avanza un espacio el cursor
      mov ah, 02h       ;petición
      mov bh, pagina    ;se pone la pagina
      int 10h                 ;avanza uno el cursor para simular el avanze de un
space despues de un tecladazo
      ret

ascii endp
;---------------------------------------------------------------------------
;Metodo que se llama cuando se presiona la tecla backspace y se encarga
;de borrar el caracter que se encuentra antes de donde esta el cursor
;---------------------------------------------------------------------------
backspace proc near

      call getcursor
      cmp dl,0h
      je dont
      add dl,-01h       ;se atraza un espacio el cursor
      mov ah, 02h       ;petición
      mov bh, pagina    ;se pone la pagina
      int 10h                 ;avanza uno el cursor para simular el avanze de un
space despues de un tecladazo
      mov aski,20h
      call ascii
      add dl,-01h       ;se atraza un espacio el cursor
      mov ah, 02h       ;petición
      mov bh, pagina    ;se pone la pagina
      int 10h                 ;se atrasa 2 veces porque el metodo ascii avanza
uno entonces solo hubiera dejado un espacio en blanco DX
dont:
      ret
```

```
backspace endp
;-------------------------------------------------------------------------
;Metodo que se llama al apachar la combinacion de teclas ctrl + c
;y graba en memoria la linea donde se encuentra el cursor
;-------------------------------------------------------------------------
copiar proc far

      mov bx, 0
      mov cx, 80

copy:
      mov al, [es:di]
      mov arreglo[bx], al
      inc bx
      call mright
      loop copy
      ret

copiar endp
;-------------------------------------------------------------------------
pegar proc near
mov bx, 0
mov cx, 80
dpaste:
mov al, arreglo[bx]
mov [es:di], al
inc bx
call mright
loop dpaste
ret
pegar endp
;-------------------------------------------------------------------------
cortar proc near
mov bx, 0
      mov cx, 80

dcut:
      mov al, [es:di]
      mov arreglo[bx], al
      mov [es:di], 720h
      inc bx
      call mright
      loop dcut
      ret
cortar endp
;-------------------------------------------------------------------------
;Cuando se presiona F1, F2, F3 o F4 estos procesos se encargan de
;cambiar de pagina de la 0 a la 3 y colocar el cursor en la fila
;0 y la columna 0
;-------------------------------------------------------------------------
f1 proc near

      mov ah, 05h  ;para cambiar de pagina
      mov al, 0   ;pagina 0
      int 10h

      mov ah, 02h
      mov bh, 0
      mov dh, 05
      mov dl, 12
      int 10h
      call getpagina
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página empieza en la 0
```

```
      mov dh, 00h       ; fila 0
      mov dl, 0h        ; columna 0 (mov dx, 050ch)
      int 10h
      mov aux,00
      call backspace
      ret

f1 endp

f2 proc near

      mov ah, 05h ;para cambiar de pagina
      mov al, 1   ;pagina 1
      int 10h

      mov ah, 02h
      mov bh, 1
      mov dh, 05
      mov dl, 12
      int 10h

      call getpagina
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página empieza en la 0
      mov dh, 00h       ; fila 0
      mov dl, 0h        ; columna 0 (mov dx, 050ch)
      int 10h



      mov aux, 00
      call backspace
      ret

f2  endp

f3 proc near

      mov ah, 05h
      mov al, 2
      int 10h

      mov ah, 02h
      mov bh, 2
      mov dh, 05
      mov dl, 12
      int 10h

      call getpagina
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página empieza en la 0
      mov dh, 00h       ; fila 0
      mov dl, 0h        ; columna 0 (mov dx, 050ch)
      int 10h

      mov aux, 00
      call backspace
      ret

f3  endp

f4 proc near

      mov ah, 05h
```

```
      mov al, 3
      int 10h

      mov ah, 02h
      mov bh, 3
      mov dh, 05
      mov dl, 12
      int 10h

      call getpagina
      mov ah, 02h       ; petición
      mov bh, pagina          ; número de página empieza en la 0
      mov dh, 00h       ; fila 0
      mov dl, 0h        ; columna 0 (mov dx, 050ch)
      int 10h
      call menu
      mov aux, 00
      call backspace
      ret

f4 endp
;-------------------------------------------------------------------------
;Este proceso se encarga de limpiar lo que se encuentra en la pagina
;actual
;-------------------------------------------------------------------------
f5 proc near

      mov ax, 0600h
      mov bh, 07h
      mov cx, 0000
      mov dx, 184fh
      int 10h
      call menu
      ret

f5 endp
;-------------------------------------------------------------------------
;Cuando se utilizan las teclas pageup y pagedown invocan a este metodo
;para poder recorrer de la pag 0 a la 3 de forma circular
;-------------------------------------------------------------------------
pages proc near
      call backspace
    cmp pag, 3
    jne change
    mov pag, -1
change:
      inc pag
      mov ah, 05h    ; petición
      mov al, pag    ; número de página
      int 10h
      mov ah, 02h
      mov bh, 3
      mov dh, 00     ; esta parte se encarga de colocar el cursor
      mov dl, 00     ; en la fila y columna 0.
      int 10h
      call backspace
    ret
pages endp
;-------------------------------------------------------------------------
;Este proceso se encarga del movimiento de las flechas izquierda y
;derecha
;-------------------------------------------------------------------------
mleft proc near
   call getcursor
```

```
   mov kursor, dx
    cmp dl, 0
    je gend
    sub di, 2          ; bajas el di primero
    dec dl
    mov ah, 02h
    mov bh, 00h
    int 10h
      gend:
      ret
endp

mright proc near
   call getcursor
   mov kursor, dx
    cmp dl, 40h
    je find
    add di, 2          ; aumenta el di
      inc dl
    mov ah, 02h
    mov bh, 00h
    int 10h
      find:
      ret
endp
;-------------------------------------------------------------------------
movup proc near

      call getcursor
      mov kursor, dx
      cmp dh,0h

      je fin2
            mov ah, 02h       ; petición
            add dh,-1h
      mov bh, pagina    ; número de página empieza en la 0

            int 10h
            add contador,-01h
fin2:
ret

movup endp
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
movdown proc near

      call getcursor
      mov kursor, dx
      cmp dh,14h

      je fin3
            mov ah, 02h       ; petición
            add dh,1h
      mov bh, pagina    ; número de página empieza en la 0

            int 10h
            add contador, 01h
fin3:
ret

movdown endp
;-------------------------------------------------------------------------
delete proc near
```

```
      mov [es:di], 720h
      ret

delete endp
;--------------------------------------------------------------------------
menudo proc far
cuatro:


menudo endp
;--------------------------------------------------------------------------

;Metodo que se encarga de verificar que caracter es el que esta entrando
;si es un tecla de funcion llama a sus respectivos metodos y asi suce-
;sivamente va comparando y si no es ninguna de las opciones quiere decir
;que es texto a ingresar
;--------------------------------------------------------------------------
insert proc far
      ; esto hace la peticion para escribir 80 caracteres por linea
      ; para con enter y llena de espacios
ingreso:
      mov ah,10h
      int 16h
      call funcion
      mov aski, al
      cmp ax, 1c0dh     ;es enter ??
      je clinea         ;si si
      cmp al,00h        ;es tecla de funcion ?
      je cfuncion       ;si si brinca a teclado extendido
      cmp al,0e0h       ;es tecla de teclado extendido?
      je ctcladoex      ;si si brinca a teclado extendido
      cmp   ax, 0f09h   ;¿es tab???
      je    tabs        ;si
      cmp ax, 0e08h   ;es backspace?
      je cback          ;si si brinka a backspace :P
      cmp ax, 2e03h ;es ctrl c
      je copis
      cmp ax, 2f16h
      je paste
      cmp ax,  2d18h
      je cut


      call ascii        ;si no es tecla especial brinca al asciii
      cmp aux, 40h      ;compara si ya llego al final de la linea adimitido
      ja clinea         ;si ya llego entonces brinca a cambio de linea simula
enter
jmp ingreso         ;si no simula de nuevo la entrada para seguir metiendo askis

clinea:
      call cambio
      ret

copis:
      call copiar
      ret
paste:
      call pegar
      ret
cut:
      call cortar
      ret
cback:
```

```
      call getpagina
      call backspace
      ret
tabs:
      call getpagina
      call tab
      ret

cfuncion:
      mov aux,0h
      mov contador,0h
      call funcion
      ret

ctcladoex:
      call tecladoextra
      ret

insert endp
;-------------------------------------------------------------------------
;Metodo que se encarga de comprobar que tecla especial se presiono
;y de acuerdo a ello se encarga de llamar a su respectivo metodo
;-------------------------------------------------------------------------
tecladoextra proc near

      cmp ax, 47e0h           ;es home ?
      je chome                ;si es home lo q presiono
      cmp ax, 4fe0h           ;es end?
      je cend                       ;si es end brinka
      cmp ax, 49e0h           ;es page up?
      je cpageup              ;si si entonces brikca
      cmp ax, 51e0h           ;es page down?
      je cpagedown            ;si es igual entonces brinca a con pagedown
      cmp ax, 4be0h           ;es flechita a la izquierda?
      je cizkierda
      cmp ax, 4de0h           ;es flechita a la derecha?
      je cderecha
      cmp ax,50e0h                            ; es flecha abajo
      je cdown
      cmp ax, 48e0h           ;es flecha arriba
      je cup

cup:
      call movup
      ret


cdown:
      call movdown
      ret

cderecha:
      call mright
      ret

cizkierda:
      call mleft
      ret

cpagedown:
      call pages
      ret

cpageup:
```

```
      call pages
      ret

chome:
      call home
      ret
cend:
      call fin

tecladoextra endp
;-------------------------------------------------------------------------
funcion proc near

            cmp ah, 3bh
            jne s1
            call f1
      s1:
            cmp ah, 3ch
            jne s2
            call f2
      s2:
            cmp ah, 3dh
            jne s3
            call f3
      s3:
            cmp ah, 3eh
            jne s4
            call f4
      s4:
            cmp ah, 3fh
            jne s5
            call f5
      s5:
            cmp ah, 41h
            je goout
            cmp ah, 01h
            je goout
            cmp ah, 49h
            jne s7
            call pages
      s7:
            cmp ah, 51h
            jne s8
            call pages
      s8:
            cmp ah, 53h
            jne s9
            call delete
      s9:
            cmp ax, 2e03h
            jne s10
            call copiar
            jmp goout
      s10:
            cmp ax, 2e10h
            jne s11
            call pegar
      s11:
            cmp ax, 2e18h
            jne s12
            call cortar
      s12:
            jmp final
goout:
```

```
call salida
final:
ret

funcion endp
;-------------------------------------------------------------------------
;Metodo que se encarga de hacer el menu
;-------------------------------------------------------------------------
menu proc near

      call getpagina
      mov ah, 05h       ; petición
      mov al, pagina    ; número de página
      int 10h

      ; hace un cuadro de color para meter las instrucciones adentro

      call menudo
      ; peticion para posicionar el cursor

      mov ah, 02h       ; petición
      mov bh, pagina    ; número de página empieza en la 0
      mov dh, 00h       ; fila 0
      mov dl, 0h        ; columna 0 (mov dx, 050ch)
      int 10h

ciclo:

      cmp contador,14h
      je exit1
      call insert
      jmp ciclo
      ret

exit1:
      mov ax,4c00h       ;salida al dos
      int 21h

menu endp
;-------------------------------------------------------------------------
;Aqui empieza el programa
;-------------------------------------------------------------------------
begin proc far

      mov ax, @data   ; inicializar area de datos
      mov ds, ax
      mov ah, 00h     ; cargo a ah 00h
      mov al, 03h     ; asigna modo texto a color 80x25
      int 10h

      call menu   ;llama al procedimiento q empieza a dibujar

salida:
      mov ah, 05h ; salida de protocolo DX
      mov al, 0
      int 10h
      mov ax, 0600h
      mov bh, 07h
      mov cx, 0000
      mov dx, 184fh
      int 10h
      mov ax,4c00h       ;salida al dos
      int 21h
      ret
```

```
begin   endp
end begin
;--------------------------------------------------------------------------
```

```
;Universidad del Valle de Guatemala
;Nery Fernando Guzman Carn, 06153
;Hugo Mota Aguilar 05129

;Ejercicio No.3
;Programa que contien unmenu contenido en tablas
;tiene 4 opciones para realizar siendo la ultima
;de salida

;-----------------------------------------------------------------
;Directivas del procesador

.MODEL SMALL
.STACK 64
; -----------------------------------------------------------------
; Inicio del area de datos
; Definicion de datos. DB = 1 byte, DW = 2 bytes

.DATA
Resul       DB 3 DUp(0) ;guarda el resultado
Termi            DB '$'
CAD1  DB 10,'El caracter leido es un número','$'
CAD2  DB 10,'El caracter leido es una letra Mayuscula','$'
CAD3  DB 10,'El caracter leido es una letra Miniscula','$'
CAD4  DB 10,'El caracter leido es un caracter especial','$'
CAD5  DB 10,'El numero es par','$'
CAD6  DB 10,'El numero es impar','$'
CAD7  DB 10,13,'EL numero al cubo es: ','$'
CAD8        DB '  ',10,13,'$'
SALIDA2      DB   10,13,   'GRACIAS POR USAR EL PROGRAMA',10,13, '$'
SALIDA      DB   10,13,   'Seleccione una de las 4 opciones',10,13
SALIDA3     DB   10,13,   'Presione 0 para elevar un digito al cubo  ',10,13
SALIDA4      DB   10,13,   'Presione 1 para  decir si un digito es numero, letra
o caracter especial ',10,13
SALIDA5      DB   10,13,   'Presione 2 para ver si es par o impar  ',10,13
SALIDA6      DB   10,13,   'Salir  ',10,13, '$'
TABLA       DW    COD0        ; tabla de direcciones
            DW    COD1
            DW    COD2
            DW    COD3
CODIGO      DB    ?

; Inicio del codigo
.CODE
;-----------------------------------------------------------------

; Funcion de entrada
;-----------------------------------------------------------------
CIN   PROC NEAR
      MOV AH, 01H    ; peticion de la funcion de entrada
      INT 21H        ; llama al DOS
      MOV CODIGO, AL ; mueve el codigo seleccionado
      SUB CODIGO, 30H; restarle el valor
      MOV Dx,0H
      LEA Dx,CAD8
      MOV AH, 09H   ; peticion para desplegar
    INT 21H       ; llama al DOS
      RET

CIN   ENDP
;-----------------------------------------------------------------
; Funcion de despliegue
;-----------------------------------------------------------------
COUT   PROC NEAR
```

```
       MOV AH, 09H   ; peticion para desplegar
       INT 21H       ; llama al DOS
       RET

COUT   ENDP
;-------------------------------------------------------------------------
;FUNCION DE SALIDA
;-------------------------------------------------------------------------
EXIT1 PROC NEAR
      MOV DX, 00H            ;Se borra DX
    LEA DX, SALIDA2     ;Se coloca en DX la direcci¢n de Centenas
    MOV AH, 09H         ;muestra el caracter en pantalla
    INT 21H
      RET

EXIT1 ENDP
;*************************************************************
;Proceso para Elevar al cubo
;*************************************************************
CU PROC FAR
    MOV AH, 07H         ;Se pide otra interrupcion en el teclado
    INT 21H

      MOV AH,0H         ;Se limpia la parte alta del registro
      SUB Al,30H        ;se le quita 30h para trabajarlo
      MOV Dl,Al         ; Se copia el numero ingresado para multi-
    MUL Dl              ;plicarlo por si mismo 2 veces
      MUL Dl
      CALL DES          ;Se llama a el metodo de despliege
      RET                     ;Regresa al procedimiento

CU ENDP
;*************************************************************
;Proceso para comparar si es letra, numero o caracter especial
;*************************************************************
COP PROC FAR

    MOV AH, 07H         ;Se pide otra interrupcion en el teclado
    INT 21H

      CMP Al,39H        ;Se compara los rangos posibles entre
      JBE NUM                 ;numero, letra minuscula y mayuscula si
      CMP Al,5AH        ;entre entre los rangos se vuelve a com-
      JBE LETM          ;parar y si no entre entre ese rango es
      CMP AL,7AH        ;caracter especial
      JBE LETI
      RET
COP ENDP
;*************************************************************
;Proceso para desplegar si es caracter especial o numero
;*************************************************************
NUM PROC FAR
      CMP Al,30H        ;Se le compara el otro rango para ver si
      JAE ccl1          ;es  numero deno serlo es caracter
      MOV DL, 00H       ;
      LEA DX, CAD4      ;Se carga el resultado de caracter

ccl2:
      MOV AH, 09H     ;Muestra los caracteres en pantalla
    INT 21H
      RET

ccl1:
      MOV DL, 00H
```

```
      LEA DX, CAD1      ;Se carga el resultado de numero
      JMP ccl2

NUM ENDP

;************************************************************
;Proceso para desplegar si es caracter especial o numero
;************************************************************
LETM PROC FAR
      CMP Al,41H        ;Se le compara el otro rango para ver si
      JAE ccl3          ;es  letra mayuscula de no serlo es caracter

      MOV DL, 00H
      LEA DX, CAD4      ;Se carga el resultado de caracter

ccl4:
      MOV AH, 09H       ;Muestra los caracteres en pantalla
    INT 21H
      RET

ccl3:
      MOV DL, 00H
      LEA DX, CAD2      ;Se carga el resultado de letra Mayuscula
      JMP ccl4

LETM ENDP

;************************************************************
;Proceso para desplegar si es caracter especial o numero
;************************************************************
LETI PROC FAR
      CMP Al,61H        ;Se le compara el otro rango para ver si
      JAE ccl5          ;es  letra minuscula de no serlo es caracter

      MOV DL, 00H
      LEA DX, CAD4      ;Se carga el arreglo del resultado

ccl6:
      MOV AH, 09H     ;Muestra los caracteres en pantalla
    INT 21H
      RET

ccl5:
      MOV DL, 00H
      LEA DX, CAD3      ;Se carga el resultado de letra Mayuscula
      JMP ccl6

LETI ENDP

;************************************************************
;Proceso para ver si es par o impar
;************************************************************
POI PROC FAR
      MOV Ah,0H         ;Se limpia la parte alta
      AND Al,01H        ;se le hace mascara
      JNZ ccl7          ;si es impar salta

      MOV DL, 00H
      LEA DX, CAD5      ;Se carga el resultado de par
ccl8:
      MOV AH, 09H     ;Se depleiga en pantalla
    INT 21H
      RET
ccl7:
```

```
      MOV DL, 00H
      LEA DX, CAD6      ;Se carga el resultado de impar
      JMP ccl8

      RET
POI ENDP

;************************************************************
;Proceso para Convertir para desplegar
;************************************************************
DES PROC FAR                  ;Se convierten las centenas y se
      MOV Bx,0                ;guardan en el arreglo
      CMP Ax, 100
      JNAE ccl9
      MOV Dl,100
      DIV Dl
      MOV Resul[Bx],Al
      MOV Dl,Ah
      MOV Al,Dl
      CBW
ccl9:
      INC Bx                        ;Se convierte las decenas y se
      CMP Ax, 10              ;guarda en el arreglo
      JNAE ccl10
      MOV Dl,10
      DIV Dl
      MOV Resul[Bx],Al
      MOV Dl,Ah
      MOV Al,Dl
      CBW
ccl10:
      INC Bx                        ;Se convierte la unidades y se
      MOV Dl,1                ;guarda en el arreglo
      DIV Dl
      MOV Resul[Bx],Al
      MOV Cx,3                ;Se inicializa un contador de ciclo
      MOV Bx,2                ;Se inicializa un indice
Ccl11:
      ADD Resul[Bx],30H ;Se le suma 30H a todas las unidades
      DEC Bx                        ;para poder desplegarlo  en pantalla.
      LOOP Ccl11
      MOV AH, 09h         ;Se despleiga la cadena en pantalla
    LEA DX, CAD7
    INT 21H

      MOV DL, 00H
      LEA DX, Resul[0]  ;Se carga el arreglo del resultado

      MOV AH, 09H         ;Muestra los caracteres en pantalla
    INT 21H
      RET

DES ENDP

;-------------------------------------------------------------------------
;Procedimiento Saltos
;-------------------------------------------------------------------------

SALTOS      PROC  NEAR

      MOV BL, CODIGO    ; obtener el codigo
      XOR   BH, BH            ; limpiar parte alta
      SHL   BX, 01            ; multiplicar el valorpor 2
      JMP   [TABLA+BX]  ; salta a la rutina de tabla
```

```
COD0:                   ;manda a llamar al proceso 1 en caso de ser elegido
      CALL  CU
      JMP   CODRET
COD1:                   ;manda a llamar al proceso COP en caso ser elegido
      CALL  COP
      JMP   CODRET
COD2:                   ;manda a llamar al proceso POI en caso de ser elegido
      CALL  POI
      JMP   CODRET
COD3:                   ;manda a llamar al proceso de salida en caso es electa.
      CALL  EXIT1
      JMP   CODRET

CODRET:     RET

SALTOS      ENDP
;-------------------------------------------------------------------------
; Procedimiento principal
;-------------------------------------------------------------------------
MAIN  PROC NEAR
      MOV AX,@data      ; inicializa segmento de datos
      MOV DS,AX
ccl12:
      MOV DX, 00H             ;Se borra DX
    LEA DX, SALIDA      ;Se coloca en DX la direcci¢n de Centenas

    MOV AH, 09H         ;muestra el caracter en pantalla
    INT 21H

      CALL CIN              ; escoge el codigo
      CALL SALTOS
      CMP CODIGO, 3
      JNE ccl12             ;mantiene el ciclo para salir

      MOV AX, 0
      MOV AH, 4CH           ; COdigo de interrupci¢n 4CH: salida al DOS
      INT 21H


MAIN  ENDP

END MAIN
```