SOLUCIONES COMENTADAS AL EXAMEN DE

LABORATORIO ESTRUCTURAS DE LOS COMPUTADORES

FEBRERO DE 1.999

(I.T.Informática de Sistemas / Gestión)

1º) Se desea realizar un fichero BAT que cumpla los siguientes requisitos:

a) Si no se le pasa ningún parámetro o se le pasa el parámetro **/?** deberá sacar un mensaje de ayuda al usuario.
b) Si se le pasa el nombre de un directorio, éste será el directorio completo del lugar en el que se encuentran los ejecutables del Microsoft Assembler.
   - El fichero deberá incluir las variables de entorno: MASM y LINK en el fichero C:\AUTOEXEC.BAT con los valores /Zi y /L para la variable MASM y el valor /CO para la variable LINK. Si las variables ya existían, se deberán eliminar las variables del C:\AUTOEXEC.BAT y asignarles el nuevo valor, tanto en el C:\AUTOEXEC.BAT como a las variables de entorno.
   - Se deberá añadir el directorio pasado como parámetro a la variable de entorno PATH y actualizarla en el fichero AUTOEXEC.BAT siempre que los ficheros MASM.EXE y LINK.EXE se encuentren en el directorio especificado como parámetro. En el caso de no existir cualquiera de los dos, el programa terminará indicando al usuario que en el directorio especificado como parámetro no se encuentran los ficheros del Microsoft Assembler 5.1
c) Si se le pasa el nombre de un fichero, sin extensión, será el nombre del fichero que se debe compilar con el MASM y posteriormente enlazar con el LINK.

Ejemplos:

C:\>ENSAMBLA

El comando ENSAMBLA compila y enlaza el fichero que se le pasa como parámetro, o actualiza el AUTOEXEC.BAT y las variables de entorno con la información del directorio que se le pasa como parámetro

La sintaxis es:

"ENSAMBLA [/? | *nombrefichero* | *nombredirectorio]"*

C:\>
C:\>ENSAMBLA C:\MASM\BIN
C:\> Las variables MASM, LINK y PATH han sido actualizadas correctamente

Solución

```
REM EVITAMOS EL ECO DE PANTALLA
@echo off

REM CONTROLAMOS QUE NO SE INTRODUZCAN PARÁMETROS O QUE SE INTRODUZCA EL
REM PARÁMETRO /? EN CUYO CASO SACAMOS LA AYUDA DEL COMANDO
IF "%1"=="" GOTO Ayuda
IF "%1"=="/?" GOTO Ayuda

REM SE COMPRUEBA SI EL PARÁMETRO ES UN DIRECTORIO
IF EXIST %1\NUL GOTO TratarDir

REM SE COMPRUEBA SI EL PARÁMETRO ES UN FICHERO DE ENSAMBLADOR
IF EXIST %1.ASM GOTO TratarArch

REM SI NO SE TERMINA CON UN ERROR
GOTO FinConErr

REM SI SE TRATA DE UN DIRECTORIO
REM ACTUALIZAMOS LAS VARIABLES MASM Y LINK EN EL ENTORNO Y EN EL
REM FICHERO AUTOEXEC.BAT
:TratarDir
SET MASM=/LI /L
SET LINK=/CO
```

```
REM COMPROBAMOS LA EXISTENCIA DEL FICHERO MASM.EXE
IF EXIST %1\MASM.EXE GOTO SeguirDir
REM SI NO EXISTE INDICAMOS UN  ERROR.
GOTO FinConErr

:SeguirDir
REM COMPROBAMOS LA EXISTENCIA DEL FICHERO LINK.EXE
REM SI NO EXISTE INDICAMOS UN ERROR.
IF NOT EXIST %1\LINK.EXE GOTO FinConErr



REM SI EXISTEN LOS DOS ACTUALIZAMOS LA VARIABLE PATH
REM ELIMINAMOS EL VALOR ANTERIOR EN EL AUTOEXEC.BAT
TYPE C:\AUTOEXEC.BAT | FIND /V /I "MASM=" >FICH
TYPE FICH | FIND /V /I "LINK=" > FICH2

REM Y ACTUALIZAMOS LOS VALORES EN EL AUTOEXEC.BAT
TYPE FICH2 | FIND /V /I "PATH=" > FICH3
ECHO PATH=%PATH%;%1 >> FICH3
ECHO SET MASM=/Zi /L >>FICH3
ECHO SET LINK=/CO >> FICH3
SET PATH=%PATH%;%1
SET MASM=/Zi /L
SET LINK=/CO

REM BORRAMOS LOS ARCHIVOS INTERMEDIOS Y ACTUALIZAMOS EL AUTOEEC.BAT
DEL FICH
DEL FICH2
COPY FICH3 C:\AUTOEXEC.BAT
DEL FICH3

REM HEMOS TERMINADO Y SALTAMOS A FIN
GOTO Fin

REM SI SE TRATA DE UN FICHERO DE ENSAMBLADOR LO COMPILAMOS
:TratarArch
MASM %1;
LINK %1;
GOTO Fin

REM MENSAJE DE AYUDA
:Ayuda
ECHO El programa compila el fichero pasado o actualiza las variables de entorno
ECHO del MS-DOS
GOTO Fin

REM MENSAJE DE ERROR.
:FinConErr
ECHO %1 no es una ruta válida o el nombre de un fichero de ensamblador correcto
:Fin
```

2º) Indica qué es lo que hace el fichero BAT siguiente sabiendo que solamente se le permiten pasar números enteros como parámetros:

```
REM DESACTIVAMOS EL ECO DE PANTALLA
@ECHO OFF

REM ENTRAMOS EN UN BUCLE
:BUCLE

REM BORRAMOS LA PANTALLA
CLS

REM AÑADIMOS EL PARÁMETRO A UN FICHERO LLAMADO FICH
ECHO %1>>FICH

REM SACAMOS POR PANTALLA CUANTAS VECES SE HA ENCONTRADO EL
REM PARÁMETRO EN EL FICHERO FICH.
type fich | find /c "%1"

REM REPETIMOS LA OPERACIÓN ANTERIOR Y BUSCAMOS EN LA SALIDA
REM SI SE ENCUENTRA EL PARÁMETRO INTRODUCIDO. SI HEMOS
REM INTRODUCIDO EL PARÁMETRO 5 Y EN FICH LO HEMOS AÑADIDO 4
REM VECES, NO LO ENCONTRARÁ. SI SON 5 VECES SI LO ENCONTRARÁ.
type fich | find /c "%1" | find "%1" >nul

REM SI NO LO ENCUENTRA, SE DEBE REPETIR EL BUCLE
if errorlevel 1 goto bucle

REM BORRAMOS EL FICHERO
del fich
:fin
```

Solución.

El programa va escribiendo los números desde el 1 hasta el valor pasado como parámetro incluido.

3º) Realizar un programa en ensamblador que lea un número hexadecimal de un máximo de cuatro cifras y lo imprima en OCTAL (base 8).

Ejemplos:

C:\> OCTAL

Introduzca un número hexadecimal de un máximo de 4 cifras, por favor: 1FF
El número convertido a octal queda: 000777

C:\> OCTAL

Introduzca un número hexadecimal de un máximo de 4 cifras, por favor: 9999
El número convertido a octal queda: 114631

Solución

Está hecho con máscaras, aunque es más fácil el ir dividiendo por la base

```
;
;*****************************************************************************
;
; A continuación definimos el segmento de datos.
;
;
;*****************************************************************************
;
DATOS SEGMENT
  ImprimeCadena EQU 09h                ; Función de la INT 21h
  LeeCadena    EQU 0Ah                 ; Función de la INT 21h
  Terminar     EQU 4Ch                 ; Función de la INT 21h
  MASCARAOCTAL EQU 7000h               ; Máscara empleada para quedarme con tres bits
  ESPACIOS      EQU 03                 ; Constante que indica cuanto debo desplazar
  DESPLAZAMIENTO DB 12                 ; Máximo número de desplazamiento
MsgDato      DB 'Introduzca un número hexadecimal de un máximo de 4 cifras, por favor: $'
  MsgEnter     DB 10,13,'$'            ; Cadena que simula el ENTER
  MsgOctal     DB 'El número convertido a octal queda: $'
  MaximoMas1   DB 5                    ; Definición de la cadena a leer
  CifrasLeidas DB 0
  Numero       DB 0, 0, 0, 0, 0
  Resultado    DB 7 DUP('$')           ; Definición de la cadena en la que se deja el resultado
DATOS ENDS
;
;*****************************************************************************
;
; A continuación definimos el segmento de pila.
;
; ***************************************************************************
;
PILA SEGMENT STACK
 DB 1024 DUP (0)
PILA ENDS
;
;*****************************************************************************
;
; A continuación definimos el segmento de código.
;
; ***************************************************************************
;
CODIGO SEGMENT 'CODE'
  ASSUME CS:CODIGO, DS:DATOS, SS:PILA
;
; ***************************************************************************
;
; A continuación definimos el procedimiento PRINCIPAL que llama a los siguientes procedimientos:
;          PedirDatos:              Solicita y lee por teclado un número de cuatro cifras hexadecimales.
;          ImprimirResultado:       Imprime el número en OCTAL.
;          TerminarPrg:             Produce el final del programa.
; Parámetros que pasa:     NINGUNO.
; Parámetros que recibe:   NINGUNO.
;
; ***************************************************************************
;
```

```
  Principal PROC FAR
  MOV AX, DATOS
  MOV DS, AX

  CALL PedirDatos               ; Pedimos el número.
  CALL ImprimirResultado        ; Imprimimos el resultado.
  CALL TerminarPrg              ; Terminamos la ejecución del programa.
  RET
  Principal ENDP
;
;*************************************************************************************
;
; PROCEDIMIENTO:    PedirDatos.
; OBJETIVOS:        Leer un número de cuatro cifras hexadecimales. Deja el número leído en
;                   la estructura MaximoMas1. Emplea la función 0Ah de la INT 21h.
; Llama a los procedimientos:
;         ImprimirCadena:     Imprime una cadena de caracteres terminada en $.
;         LeerCadena:         Lee una cadena de caracteres.
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  NINGUNO.
;
; ***********************************************************************************
;
  PedirDatos PROC NEAR
  PUSH DX                       ; Guardamos el registro que vamos a emplear.
  LEA DX, MsgDato               ; Ponemos en DS:DX la dirección de la frase
  CALL ImprimirCadena           ; Imprimimos la cadena
  LEA DX, MaximoMas1            ; Ponemos en DS:DX la dirección de la cadena a leer.
  CALL LeerCadena               ; Leemos la cadena.
  POP DX                        ; Recuperamos el valor del registro usado.
  RET                           ; Volvemos al procedimiento que nos llamó.
  PedirDatos ENDP
;
;*************************************************************************************
;
; PROCEDIMIENTO:    ImprimirCadena.
; OBJETIVOS:        Imprime una cadena de caracteres terminada en $ por medio de la función 09h
;                   de la INT 21h.
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  La  dirección de la cadena a imprimir en DS:DX.
;
; ***********************************************************************************
;
  ImprimirCadena PROC NEAR
  PUSH AX                       ; Guardamos el registro que vamos a emplear.
  MOV AH, ImprimeCadena         ; Imprimimos la cadena mediante la función 09h
  INT 21h                       ; de la INT 21h.
  POP AX                        ; Recuperamos el valor del registro usado.
  RET                           ; Volvemos al procedimiento que nos llamó.
  ImprimirCadena ENDP

  LeerCadena PROC NEAR
  PUSH AX                       ; Guardamos el registro que vamos a emplear.
  MOV AH, LeeCadena             ; Leemos la cadena mediante la función 0Ah
  INT 21h                       ; de la INT 21h.
  POP AX                        ; Recuperamos el valor del registro usado.
  RET                           ; Volvemos al procedimiento que nos llamó.
  LeerCadena ENDP
```

```
;
;*****************************************************************************
;
; PROCEDIMIENTO:     TerminarPrg.
; OBJETIVOS:         Termina la ejecución del programa en curso mediante la función 4Ch
;                    de la INT 21h.
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  La  dirección de la cadena a imprimir en DS:DX.
;
; *****************************************************************************
;
 TerminarPrg PROC NEAR
  PUSH AX           ; Guardamos el registro que vamos a usar.
  MOV AH, Terminar  ; Terminamos la ejecución del programa mediante la función 4Ch
  INT 21h           ; de la INT 21h
  POP AX            ; Recuperamos el valor del registro usado.
  RET
  TerminarPrg ENDP
;
;*****************************************************************************
;
; PROCEDIMIENTO:     ImprimiResultado.
; OBJETIVOS:         Convierte el número en OCTAL e imprime el resultado. Llama a los
;                    procedimientos siguientes:
;
;          ConvertirCadena: convierte la cadena de código ASCII a valores numéricos.
;          JuntarCifras:    junta los valores numéricos en una número de cuatro cifras hexadecimales.
;          ConvertirAOctal: convierte el número en OCTAL.
;          Imprimir:        imprime el número OCTAL
;
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  NINGUNO.
;
; *****************************************************************************
;
  ImprimirResultado PROC NEAR
  CALL ConvertirCadena      ; Convertimos en número el código ASCII.
  CALL JuntarCifras         ; Juntamos las cifras en un número
  CALL ConvertirAOctal      ; Convertimos el número a OCTAL.
  CALL Imprimir             ; Imprimimos el número OCTAL.
  RET                       ; Volvemos al procedimiento que nos llamó.
  ImprimirResultado ENDP
;
;*****************************************************************************
;
; PROCEDIMIENTO:     ConvertirCadena.
; OBJETIVOS:         Convierte la cadena leída (caracteres ASCII) en sus correspondientes dígitos
;                    numéricos. Llama al procedimiento:
;          ConvertirANumero:    convierte en valor numérico el ASCII que le pasamos en AL.
;
; Parámetros que pasa:    AL  con el código ASCII a convertir.
; Parámetros que recibe:  NINGUNO.
;
; *****************************************************************************
;
  ConvertirCadena PROC NEAR
  PUSH BX           ; Guardamos los registros que vamos a emplear
  PUSH CX
  XOR CX, CX        ; Ponemos a cero el registro contador.
  MOV CL, CifrasLeidas ; Ponemos en CL cuantas cifras hemos leído
```

```
  LEA BX, Numero        ; Ponemos en DS:BX la dirección donde se almacena la cadena de números.
  Bucle:
    MOV AL, [BX]                ; Llevamos a AL el carácter ASCII.
    CALL ConvertirANumero       ; Convertimos el ASCII en número.
    MOV [BX], AL                ; Cambiamos el carácter ASCII por el número calculado.
    INC BX                      ; Apuntamos al siguiente carácter.
    LOOP Bucle                  ; Decrementamos CX en uno y si  no es cero volvemos al bucle.
  POP CX                        ; Recuperamos los valores de los registros que hemos empleado
  POP BX
  RET                           ; Volvemos al procedimiento que nos llamó.
  ConvertirCadena ENDP
;
;*******************************************************************************
;
; PROCEDIMIENTO:    ConvertirANumero.
; OBJETIVOS:        Convierte número el código ASCII que le pasamos:
;
; Parámetros que devuelve:      Devuelve en AL el valor numérico del ASCII recibido.
; Parámetros que recibe:        En AL el código ASCII a convertir.
;
; ******************************************************************************
;
  ConvertirANumero PROC NEAR
    CMP AL, '9'         ; Comparamos con el código ASCII del 9
    JLE EsNumero        ; Si es menor, se tratará de una cifra entre 0 y 9 se restará 30h
    SUB AL, 7H          ; SI no, es una cifra entre A y F, por lo que se debe restar 37h. Primero
                        ; restamos 7h.
  EsNumero:
    SUB AL, 30h         ; Restamos 30h
    RET                 ; Volvemos al procedimiento que nos llamó.
  ConvertirANumero ENDP
;
;*******************************************************************************
;
; PROCEDIMIENTO:    JuntarCifras.
; OBJETIVOS:        Junta todas las cifras de la cadena leída en
;
;       ConvertirCadena: convierte la cadena de código ASCII a valores numéricos.
;       JuntarCifras:    junta los valores numéricos en una número de cuatro cifras hexadecimales.
;       ConvertirAOctal: convierte el número en OCTAL.
;       Imprimir:        imprime el número OCTAL
;
; Parámetros que devuelve:      AX con el número hexadecimal.
; Parámetros que recibe:        NINGUNO.
;
; ******************************************************************************
;
   JuntarCifras PROC NEAR
  PUSH BX               ; Guardamos el valor de los registros que vamos a emplear.
  PUSH CX
  PUSH DX
  XOR AX, AX            ; Ponemos a cero los registros AX, CX y DX
  XOR CX, CX
  XOR DX, DX
  MOV CL, CifrasLeidas  ; Ponemos en CL el número de cifras leídas.
  DEC CL                ; Decrementamos en una unidad ese número de cifras.
  LEA BX, Numero        ; Ponemos en DS:BX la dirección de los números.
  BucleJuntar:
    MOV DL, [BX]        ; Ponemos en DL el número
    ADD AX, DX          ; Sumamos AX y DX.
```

```
    SHL AX, 1           ; Desplazamos AX cuatro veces (una cifra hexadecimal).
    SHL AX, 1
    SHL AX, 1
    SHL AX, 1
    INC BX              ; Incrementamos BX para apuntar al siguiente número.
    LOOP BucleJuntar    ; Decrementa CX en 1 y si no es cero salta a BucleJuntar.
  MOV DL, [BX]          ; Ponemos en DL el último número.
  ADD AX, DX            ; Suma la última cifra a AX
  POP DX                ; Recuperamos el valor de los registros.
  POP CX
  POP BX
  RET                   ; Volvemos al procedimiento que nos llamó.
  JuntarCifras ENDP
;
;*****************************************************************************************
;
; PROCEDIMIENTO:    ConvertirAOctal.
; OBJETIVOS:        Convierte a OCTAL el número recibido en AX

; Parámetros que devuelve:      NINGUNO.
; Parámetros que recibe:        AX con el número hexadecimal.
;
; *****************************************************************************************
;
  ConvertirAOctal PROC NEAR
  PUSH BX               ; Guardamos los registros que vamos a emplear.
  PUSH CX
  PUSH DX
  XOR DX, DX            ; Ponemos a cero DX
  MOV BX, AX            ; Copiamos en AX el valor de AX.
  AND BX, 8000h         ; Verificamos si el primer dígito octal será cero o uno. 16 bits hexadecimales
                        ; Son 6 dígitos octales, ya que se agrupan de derecha a izquierda de 3 en tres.
  JZ GuardaCero         ; Si es cero guardaremos el ASCII del cero
  MOV DL, 31h           ; Si no es cero, será uno con lo que guardamos el ASCII del uno.
  JMP Seguir
  GuardaCero:
  MOV DL, 30h           ; Guardamos el ASCII del cero.
  Seguir:
  LEA SI, Resultado     ; Ponemos en DX:SI la dirección donde almacenaremos el resultado.
  MOV [SI], DL          ; Ponemos como primer dígito OCTAL el valor calculado.
  XOR CX, CX            ; Ponemos a cero CX.
  MOV CL, 5             ; Repetiremos el bucle 5 veces, ya que quedan 5 dígitos octales.
  MOV DX, MASCARAOCTAL ; Llevamos a DX la máscara OCTAL.
  BucleOctal:
    MOV BX, AX          ; Sacamos copia de AX en BX.
    AND BX, DX          ; Realizamos un AND De BX con la 7000h
    PUSH CX             ; Guardamos en la pila CX
    MOV CL, DESPLAZAMIENTO ; Ponemos en CL el valor desplazamiento 12
    SHR BX, CL          ; Desplazamos BX 12 posiciones a la derecha con lo que obtenemos
                        ; la segunda cifra OCTAL.
    INC SI              ; Avanzamos en la cadena resultado una posición.
    ADD BL, 30h         ; Convertimos en ASCII el valor del dígito.
    MOV [SI], BL        ; Guardamos la segunda cifra.
    MOV CL, ESPACIOS ; Ponemos en CL, espacios (3)
    SHR DX, CL          ; Desplazamos la máscara 3 posiciones a la derecha.
    MOV CL, DESPLAZAMIENTO ; Ponemos en CL desplazamiento (12, 9, 6, 3)
    SUB CL, ESPACIOS            ; Le restamos espacios (3)
    MOV DESPLAZAMIENTO, CL  ; El nuevo desplazamiento es el calculado para la siguiente cifra.
    POP CX                      ; Recupero el calor de CX
    LOOP BucleOctal             ; Decremento CX en uno, si no es cero salto a BucleOctal.
```

```
  POP DX                ; Recupero el valor de los registros.
  POP CX
  POP BX
  RET                   ; Llamo al procedimiento que nos llamó.
  ConvertirAOctal ENDP
;
;*****************************************************************************
;
; PROCEDIMIENTO:    Imprimir.
; OBJETIVOS:        Imprime el resultado, llama al procedimiento
;
;       ImprimirCadena:     Imprime una cadena de caracteres terminado en $ mediante la función
;                           09h de la INT 21h
;
; Parámetros que pasa:      En DS:DX las direcciones de las cadenas a imprimir.
; Parámetros que recibe:    NINGUNO.
;
; *****************************************************************************
;
  Imprimir PROC NEAR
  PUSH DX               ; Guardo el contenido del registro que voy a usar.
  LEA DX, MsgEnter      ; Pongo en DS:DX la dirección de la cadena MsgEnter.
  CALL ImPrimirCadena   ; Imprimo la cadena.
  LEA DX, MsgOctal      ; Pongo en DS:DX la dirección de la cadena MsgOctal.
  CALL ImprimirCadena   ; Imprimo la cadena.
  LEA DX, Resultado     ; Pongo en DS:DX la dirección de la cadena Resultado.
  CALL ImprimirCadena   ; Imprimo la cadena.
  POP DX                ; Recupero el contenido de DX.
  RET                   ; Volvemos al procedimiento que nos llamó.
  Imprimir ENDP
codigo ends
end principal
```

4º) Realizar un programa en ensamblador que lea una cadena de un máximo de 8 caracteres y presente el siguiente menú:

1.- Crear el directorio
2.- Borrar el directorio
3.- Cambiar al directorio
4.- Salir

En el caso de que no sea ninguno de las opciones anteriores, el programa deberá imprimir el código ASCII 07 por pantalla de forma que se produzca un pitido.

Las funciones para crear, borrar y cambiar a un directorio son 39h, 3Ah y 3Bh respectivamente. Todas pertenecen a la interrupción 21h. Requieren que se les pase como parámetro la cadena con el nombre del directorio en DS:DX.

Se pide que el programa cree, borre o cambie al directorio que se corresponde con la cadena que se ha introducido.

Si se produce un error la función activa el flag de Acarreo, por lo que si existe acarreo se deberá mostrar el mensaje “Directorio erróneo”

Para ello el programa deberá contener al menos:

- Un procedimiento que lea una cadena de como mucho 8 caracteres.
- Un procedimiento que imprima una cadena de caracteres terminada en $.
- Un procedimiento que copie la cadena introducida menos el enter en otra cadena definida con un máximo de 8 caracteres.
- Más los procedimientos que se crean necesarios para escribir el menú y comprobar la opción introducida.

Ejemplos:

C:\> DIRECTO

Introduce el nombre de un directorio
juegos

1.- Crear directorio 2.- Borrar directorio 3.- Cambiar al directorio 4.- Salir 1

C:\> DIRECTO

Introduce el nombre de un directorio
juegos

1.- Crear directorio 2.- Borrar directorio 3.- Cambiar al directorio 4.- Salir 1
Directorio erróneo (*no lo puede crear porque ya existe, lo creamos en el otro ejemplo*)

Solución.

```
;
;*****************************************************************************
;
; A continuación definimos el segmento de datos.
;
;
;*****************************************************************************
;
datos SEGMENT
   Menu              DB '1.- Crear directorio ', 10, 13          ;Texto del menu
                     DB '2.- Borrar directorio ', 10, 13
                     DB '3.- Cambiar al directorio ', 10, 13
                     DB '4.- Salir $'
  MsgInt             DB 'Introduce el nombre de un directorio',10,13,10,13,'$'
  MsgErr             DB 'Directorio erróneo $'
  Enter              DB 10,13,'$'                    ; Cadena que simula el Enter.
  Terminar           EQU 4Ch                         ; Función de la INT 21h.
  LeeCadena          EQU 0Ah                         ; Función de la INT 21h.
  ImprimeCadena      EQU 09h                 ; Función de la INT 21h.
  Cadena             DB 9                    ; Estructura de la cadena a leer.
                     DB 0
  Str                DB 9 dup (0)
  Res                DB 8 DUP (0)            ; Directorio sin el ENTER.
datos ENDS
;
;*****************************************************************************
;
; A continuación definimos el segmento de pila.
;
; ***************************************************************************
;
pila SEGMENT STACK
 DB 1024 DUP(0)
pila ENDS
```

```
;
;*****************************************************************************
;
; A continuación definimos el segmento de código.
;
; *****************************************************************************
;
CODIGO SEGMENT 'CODE'
  ASSUME CS:CODIGO, DS:DATOS, SS:PILA
;
; *****************************************************************************
;
; A continuación definimos el procedimiento PRINCIPAL que llama a los siguientes procedimientos:
;       PedirDirectorio:        Solicita y lee por teclado el directorio.
;       QuitarEnter:            Elimina el enter del final de la cadena.
;       PresentarMenu:          Imprime el menú de opciones.
;       TerminarPrg:            Produce el final del programa.
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  NINGUNO.
;
; *****************************************************************************
;
  ASSUME CS:codigo, DS:datos, SS:pila
Principal PROC FAR
        MOV AX, DATOS
        MOV DS, AX

        CALL PedirDirectorio
        CALL QuitarEnter
        CALL PresentarMenu
        CALL TerminarPrg
         RET
Principal ENDP
;
;*****************************************************************************
;
; PROCEDIMIENTO:    PedirDirectorio.
; OBJETIVOS:        Solicita el nombre del directorio por teclado. Llama al procedimiento
;
;       ImprimirCadena:         imprime una cadena de caracteres terminada en $ mediante la función
;                               09h de la INT 21h.
;
; Parámetros que pasa:    En DS:DX las direcciones de la cadena a imprimir.
; Parámetros que recibe:  NINGUNO.
;
; *****************************************************************************
;
PedirDirectorio PROC NEAR
        LEA DX, msgInt          ; Ponemos en DS:DX la dirección de la cadena a escribir.
        CALL ImprimirCadena     ; Llamamos a ImprimirCadena.
        LEA DX, cadena          ; Ponemos en DS:DX la dirección de la cadena Cadena
        MOV AH, LeeCadena       ; Función 0Ah de la INT 21h
        INT 21h
        RET                     ; Volvemos al procedimiento que nos llamó.
PedirDirectorio ENDP
```

```
;
;*****************************************************************************
;
; PROCEDIMIENTO:     QuitarEnter.
; OBJETIVOS:         Elimina el Enter de la cadena de entrada Str y lo deja en la cadena Res.
;
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  NINGUNO.
;
; ****************************************************************************
;
QuitarEnter PROC NEAR
        PUSH CX             ; Guardamos los registros que vamos a usar..
        PUSH DX
        XOR CX, CX          ; Ponemos a cero CX.
        MOV CL, Cadena[1]   ; Ponemos en CL el número de caracteres leídos.
        LEA BX, Str         ; Ponemos en DS:DX la dirección donde comienza el directorio.
        LEA SI, Res         ; Ponemos en DS:DX la dirección donde dejar el resultado.
        BUCLE:
                MOV DL, [BX]  ; Llevamos a DL el contenido de la dirección apuntada por BX
                MOV [SI], DL  ; Llevamos a SI el carácter del directorio
                 INC SI       ; Apuntamos al sitio en el que dejar el carácter siguiente.
                 INC BX       ; Apuntamos al carácter siguiente.
                 LOOP BUCLE
        POP DX              ; Recuperamos el contenido de los registros usados.
        POP CX
        RET                 ; Volvemos al procedimiento que os llamó.
QuitarEnter ENDP
;
;*****************************************************************************
;
; PROCEDIMIENTO:     PresentarMenu.
; OBJETIVOS:         Presenta el menú de opciones ce crear, borrar, cambiar al y salir y ejecuta la
;                    acción solicitada. Llama al procedimiento:
;
;       ImprimirCadena:     imprime una cadena de caracteres terminada en $ mediante la función
;                           09h de la INT 21h.
;
; Parámetros que pasa:    En DS:DX las direcciones de la cadena a imprimir.
; Parámetros que recibe:  NINGUNO.
;
; ****************************************************************************
;
PresentarMenu PROC NEAR
        PUSH DX             ; Guardamos en la pila el contenido de los registros que vamos a usar.
        PUSH AX
        LEA DX, Enter       ; Ponemos en DS:DX la dirección de comienzo de la cadena Enter
        CALL ImprimirCadena ; Imprimimos la cadena mediante la función 09h de la INT 21h
        LEA DX, Menu        ; Ponemos en DS:DX la dirección de comienzo de la cadena Menu.
        CALL ImprimirCadena ; Imprimimos la cadena mediante la función 09h de la INT 21h
        LEA DX, Res         ; Ponemos en DS:DX la dirección de comienzo del nombre del
                            ; directorio.
        Pedir:
                MOV AH, 01  ; Función 01h de la INT 21h (lee un carácter por teclado)
                INT 21H
                CMP AL, '1' ; Comparo AL con 1
                JE Crear    ; Si es 1 debo crear el directorio.
                CMP AL, '2' ; Comparo AL con 2
                JE Borrar   ; Si es 1 debo borrar el directorio.
                CMP AL, '3' ; Comparo AL con 3
```

```
                JE Cambiar      ; Si es 1 debo cambiar al directorio.
                CMP AL, '4'     ; Comparo AL con 4
                JE Salir        ; Si es 1 debo salir del programa.
                MOV AH, 02      ; Función 02h de la INT 21h (escribir un carácter por pantalla)
                MOV DL, 07      ; Ponemos en DL el código ASCII del pitido.
                INT 21H
                JMP Pedir       ; Volvemos a pedir un carácter.
        Crear:
                MOV AH, 39H     ; Función 39h de la INT 21h (crear directorio apuntado por DS:DX)
                INT 21H
                JC Error        ; Si hay acarreo el directorio no era válido
                JMP Salir       ; Si está bien salimos del programa.
        Borrar:
                MOV AH, 3AH     ; Función 3Ah de la INT 21h (borrar directorio apuntado por DS:DX)
                INT 21H
                JC Error        ; Si hay acarreo el directorio no era válido
                JMP Salir       ; Si está bien salimos del programa.
        Cambiar:
                MOV AH, 3BH     ; Función 3Bh de la INT 21h (cambiar al directorio apuntado por
                                ; DS:DX)
                INT 21H
                JNC Salir       ; Si está bien salimos del programa.
        Error:
                LEA DX, MsgErr          ; Ponemos en DS:DX el  mensaje de error.
                CALL ImprimirCadena     ; Imprimimos la cadena mediante la función 09h
                                        ; de la INT 21h
        SALIR:
                POP AX          ; Recuperamos el valor de los registros empleados
                POP DX
                RET             ; Volvemos al procedimiento que nos llamó.
PresentarMenu ENDP
;
;*******************************************************************************
;
; PROCEDIMIENTO:        ImprimirCadena.
; OBJETIVOS:            Imprime una cadena de caracteres terminada en $ por medio de la función 09h
;                       de la INT 21h.
; Parámetros que pasa:  NINGUNO.
; Parámetros que recibe: La  dirección de la cadena a imprimir en DS:DX.
;
; ******************************************************************************
;
  ImprimirCadena PROC NEAR
  PUSH AX                       ; Guardamos el registro que vamos a emplear.
  MOV AH, ImprimeCadena         ; Imprimimos la cadena mediante la función 09h
  INT 21h                       ; de la INT 21h.
  POP AX                        ; Recuperamos el valor del registro usado.
  RET                           ; Volvemos al procedimiento que nos llamó.
  ImprimirCadena ENDP

  LeerCadena PROC NEAR
  PUSH AX                       ; Guardamos el registro que vamos a emplear.
  MOV AH, LeeCadena             ; Leemos la cadena mediante la función 0Ah
  INT 21h                       ; de la INT 21h.
  POP AX                        ; Recuperamos el valor del registro usado.
  RET                           ; Volvemos al procedimiento que nos llamó.
  LeerCadena ENDP
```

```
 ;
;*****************************************************************************
;
; PROCEDIMIENTO:      TerminarPrg.
; OBJETIVOS:          Termina la ejecución del programa en curso mediante la función 4Ch
;                     de la INT 21h.
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  La  dirección de la cadena a imprimir en DS:DX.
;
; *****************************************************************************
;
 TerminarPrg PROC NEAR
   PUSH AX              ; Guardamos el registro que vamos a usar.
   MOV AH, Terminar     ; Terminamos la ejecución del programa mediante la función 4Ch
   INT 21h              ; de la INT 21h
   POP AX               ; Recuperamos el valor del registro usado.
   RET
   TerminarPrg ENDP

codigo ends
end principal
```