SOLUCIONES COMENTADAS AL EXAMEN DE LABORATORIO ESTRUCTURAS DE LOS COMPUTADORES FEBRERO DE 1.998

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 cualquier otro parámetro deberá tomarlo como las iniciales de aquellos ficheros que deberá eliminar del directorio actual.
c) Se le pueden pasar varias iniciales diferentes.

Ejemplos:

C:\>DELINI

El comando DELINI elimina del directorio actual todos los ficheros que coincidan las primeras letras del nombre.

La sintaxis es:

"DELINI [/?| iniciales1 [iniciales2 [iniciales3 [...]]]]"

C:\>
C:\>DELINI 1 2 3
C:\>

SOLUCIÓN:

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

REM MIRAMOS SI NO SE LE PASAN PARÁMETROS O SI EL PARÁMETRO ES EL /?
REM EN CUYO CASO MOSTRAREMOS LA AYU DA DEL COMANDO
IF "%1"=="" GOTO Ayuda
IF "%1"=="/?" GOTO Ayuda

REM INICIALIZAMOS LA VARIABLE CONDICIÓN CON EL PRIMER PÁRAMETRO PASADO.
SET Condicion=%1*.*

REM REALIZAMOS UN PARA RECORRER TODOS LOS PARÁMETROS Y ACTUALIZAMOS
REM LA VARIABLE Condicion
:BucleCondicion

REM MIRAMOS EL SIGUIENTE PARÁMETRO, Y SI NO HAY MÁS PARÁMETROS
REM PROCEDEMOS A ELIMINAR LOS FICHEROS.
SHIFT
  IF "%1"=="" GOTO Eliminar

REM ACTUALIZAMOS LA VARIABLE Condicion CON LAS SIGUIENTES INICIALES
  SET Condicion=%Condicion% %1*.*
  GOTO BucleCondicion

REM MOSTRAMOS LA AYUDA OARA EL COMANDO.
:Ayuda
  ECHO.
  ECHO El comando DELINI borra todos los ficheros que coincidan
  ECHO con las primeras letras del nombre.
  ECHO.
  ECHO La sintaxis es:
  ECHO.
  ECHO "DELINI [/?| iniciales1 [iniciales2 [iniciales3 [ ... ] ] ] ]"
  GOTO Fin
```

```
REM ELIMINAMOS LOS FICHEROS CON UN FOR
:Eliminar
  FOR %%F IN (%Condicion%) DO DEL %%F

REM CUANDO HEMOS TERMINADO BORRAMOS LA VARIABLE Condicion
:Fin
  SET Condicion=
```

2º) Se desea realizar un fichero BAT que cumpla los siguientes requisitos:

a) Tome el primer parámetro como directorio al que se le deben copiar todos los ficheros que vienen a continuación, hasta que no haya más parámetros o aparezca el parámetro /OTRODIR
b) El siguiente parámetro al parámetro /OTRODIR se tomará entonces como nuevo directorio al que copiar todos los ficheros que vengan a continuación.
c) El proceso seguirá hasta que no existan más parámetros.

Ejemplos:

C:\>COPIADIR

El comando COPIADIR copia al directorio especificado todos los ficheros que vienen a continuación. Si se encuentra el parámetro /OTRODIR copiará al parámetro que le siga los ficheros que tenga a continuación, bien hasta que no haya más parámetros o hasta que encuentre otro /OTRODIR en cuyo caso hará lo mismo

La sintaxis es:

"COPIADIR [/?| Dir1 fch1 [fch2 [... [/OTRODIR Dir2 fch1 [Fch 2 [...]]]]]]"

C:\>COPIADIR c:\tmp f1.txt f2.txt /OTRODIR c:\dos f1.com f2.com f3.exe

SOLUCIÓN:

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

REM MIRAMOS SI NO SE LE PASAN PARÁMETROS O SI EL PARÁMETRO ES EL /?
REM EN CUYO CASO MOSTRAREMOS LA AYUDA DEL COMANDO
IF "%1"=="" GOTO Ayuda
IF "%1"=="/?" GOTO Ayuda

REM SI NO EXISTE UN SEGUNDO PARÁMETRO AVISAREMOS DE QUE EXISTE UN ERROR.
IF "%2"=="" GOTO ErrorFch

REM CREAMOS LA VARIABLE Directorio QUE CONTENDRÁ EL DIRECTORIO AL QUE
REM COPIAR LOS ARCHIVOS
SET Directorio=%1

REM ENTRAMOS EN EL BUCLE PARA COPIAR LOS ARCHIVOS
REM EN EL COMPROBAMOS QUE EL DIRECTORIO ESPECIFICADO EXISTE, EN CASO
REM DE QUE NO EXISTA, INDICAREMOS EL ERROR.
REM ADEMÁS ANTES DE COPIAR UN FICHERO COMPROBAMOS QUE EXISTE.
:BucleCopiar
  IF NOT EXIST %Directorio%\NUL GOTO ErrorDir
  IF EXIST %2 COPY %2 %Directorio% /Y
```

```
REM MIRAMOS EL SIGUIENTE PARÁMETRO, Y SI NO HAY MÁS PARÁMETROS
REM HEMOS TERMINADO.
REM SI EL PARÁMETRO ES /OTRODIR, TENDREMOS QUE CAMBIAR AL DIRECTORIO
REM DESTINO AL QUE COPIAR LOS FICHEROS SIGUIENTES.
SHIFT
  IF "%2"=="" GOTO Fin
  IF "%2"=="/OTRODIR" GOTO OtroBucle
  GOTO BucleCopiar

REM DESPLAZAMOS DOS VECES LOS PARÁMETROS A LA IZQUIERDA CON LO QUE
REM EL NUEVO NOMBRE DEL DIRECTORIO ESTÁRA EN LA VARIABLE %1.
REM ACTUALIZAREMOS LA VARIABLE Directorio CON ESA VARIABLE.
REM Y VOLVEMOS A REALIZAR EL BUCLE DE COPIA.
:OtroBucle
  SHIFT
  SHIFT
  SET Directorio=%1
  GOTO BucleCopiar

REM MOSTRAREMOS EL MENSAJE DE ERROR DE QUE NO EXISTE EL FICHERO Y
REM TERMINAREMOS LA EJECUCIÓN DEL PROGRAMA.
:ErrorFch
  ECHO No se ha especificado el fichero a copiar en %Directorio%
  GOTO Fin

REM INDICAREMOS QUE EL DIRECTORIO ESPECIFICADO NO EXISTE Y TERMINAREMOS
REM LA EJECUCIÓN DEL PROGRAMA.
:ErrorDir
  ECHO El directorio %Directorio% especificado no existe
  GOTO Fin

REM MOSTRAREMOS LA AYUDA DEL COMANDO
:Ayuda
  ECHO.
  ECHO El comando COPIADIR copia al directorio especifiado
  ECHO todos los ficheros que vienen a continuacion. Si se
  ECHO encuetra el parámetro /OTRODIR copiará al parámetro
  ECHO que le siga los ficheros que tengaa continuació, bien
  ECHO hasta que no haya más parámetros o hasta que encuentre
  ECHO otro /OTRODIR en cuyo caso hará lo mismo
  ECHO.
  ECHO La sintaxis es:
  ECHO.
  ECHO "COPIADIR [/?| Dir1 fch1 [fch2 [ ... [/OTRODIR Dir2 fch1 [Fch 2 [...]]]]]"
  GOTO Fin

REM ELIMINAMOS LOS FICHEROS MEDIANTE UN FOR.
:Eliminar
  FOR %%F IN (%Condicion%) DO DEL %%F

REM TERMINAREMOS EL PROGRAMA Y ELIMINAMOS LA VARIABLE Directorio
:Fin
  SET Directorio=
```

3º) Se desea realizar un fichero BAT que busque una palabra (el primer parámetro) en todos los ficheros que vengan a continuación. Debe indicar si se ha encontrado la palabra en TODOS los ficheros o si no se ha encontrado en TODOS los ficheros

Ejemplos:

C:\> FINDIN

El comando FINDIN busca una palabra en todos los ficheros que vengan a continuación.
La sintaxis es:

FINDIN [/?| Cadena Fichero1 [Fichero2 [Fichero3 [...]]]]"

C:\> FINDIN @echo 1.bat 2.bat 3.bat 4.bat

La palabra ha sido encontrada en todos los ficheros.

SOLUCIÓN:

```
REM DESACTIVAREMOS EL ECHO DE PANTALLA
@ECHO OFF


REM MIRAMOS SI NO SE LE PASAN PARÁMETROS O SI EL PARÁMETRO ES EL /?
REM EN CUYO CASO MOSTRAREMOS LA AYU DA DEL COMANDO
IF "%1"=="" GOTO Ayuda
IF "%1"=="/?" GOTO Ayuda

REM INICIALIZAMOS LA VARIABLE Cadena CON EL CONTENIDO DE LO QUE BUSCAMOS
REM ENTRE COMILLAS DOBLES, PARA NO TENER QUE PONERLAS EN EL COMANDO FIND.
SET Cadena="%1"

REM ENTRAMOS EN EL BUCLE DE BUSCAR. SI NO ENCONTRAMOS LA CADENA EN EL
REM ARCHIVO, COMO DEBE ENCONTRARSE EN TODOS, INDICAREMOS QUE LA CADENA
REM NO SE ENCUENTRA EN TODOS LOS ARCHIVOS.

:BucleBuscar
  FIND /I %Cadena% %2 >NUL
  IF ERRORLEVEL 1 GOTO NoEsta

REM ACTUALIZAMOS LOS PARÁMETROS. SI YA NO QUEDAN MÁS ARCHIVOS EN
REM LOS QUE BUSCAR, SERÁ QUE SI QUE SE HA ENCONTRADO.
  SHIFT
  IF "%2"=="" GOTO SiEsta
  GOTO BucleBuscar

REM SACAMOS LA AYUDA DEL COMANDO POR PANTALLA
:Ayuda
  ECHO.
  ECHO El comando FINDIN busca una palabra en todos los
  ECHO ficheros que vengan a continuación.
  ECHO.
  ECHO La sintaxis es:
  ECHO.
  ECHO "FINDIN [/?| Cadena Fichero1 [Fichero2 [Fichero3 [ ... ] ] ] ]"
  GOTO Fin

REM VISUALIZAMOS QUE LA CADENA NO ESTÁ EN TODOS LOS ARCHIVOS
REM Y TERMINAMOS LA EJECUCIÓN DEL PROGRAMA.
:NoEsta
  ECHO La palabra no se encuentra en todos los ficheros.
  GOTO Fin
```

```
REM VISUALIZAMOS QUE LA CADENA ESTÁ EN TODOS LOS ARCHIVOS
REM Y TERMINAMOS LA EJECUCIÓN DEL PROGRAMA.
:SiEsta
  ECHO La palabra ha sido encontrada en todos los ficheros.

REM TERMINAMOS LA EJECUCIÓN DEL PROGRAMA ELIMINANDO LA
REM VARIABLE Cadena
:Fin
  SET Cadena=
```

4º) Realizar un programa en ensamblador que lea una cadena máxima de 80 números entre el 0 y el 9 y la imprima de forma comprimida. La forma de compresión es juntar cada dos cifras en una. Si el número de cifras introducidas es impar la última no es necesario comprimirla.

Ejemplos:

C:\> COMPRIME

Introduzca una cadena de números entre 0 y 9 por favor
23345
La cadena comprimida (algunos caracteres no son imprimibles) será:
#4]

C:\> COMPRIME

Introduzca una cadena de números entre 0 y 9 por favor
456789
La cadena comprimida (algunos caracteres no son imprimibles) será:
Egë

SOLUCIÓN:

```
;
; ********************************************************************************
;
; A continuación definimos el segmento de datos.
;
; ********************************************************************************
;
DATOS SEGMENT
    TerminarPrg       EQU   4C00h              ; Función de la INT 21h
    CorregirNumero    EQU   30h
    EscribeCadena     EQU   09h                ; Función de la INT 21h
    LeeCadena         EQU   0Ah                ; Función de la INT 21h
    MsgPedirFrase      DB   'Introduzca una cadena de numeros entre 0 y 9 por favor', 10, 13, '$'
    MsgResultado  DB   'La cadena comprimida (algunos caracteres no son imprimibles) será: ', 10, 13, '$'
    MsgEnter           DB   10, 13, '$'         ; Cadena de texto que simula un ENTER
    Frase              DB   81, 0               ; Estrucutra para poder leer una cadena de 80
                       DB   81 DUP (0)          ; caracteres mediante la función 0Ah de la INT 21h.
    FraseComprimida    DB   41 DUP (0)          ; Estrucutra para dejar la cadena comprimida.
                       DB   '$'
DATOS ENDS
```

```
;
; ********************************************************************************************
;
; A continuación definimos el segmento de pila.
;
; ********************************************************************************************
;
PILA SEGMENT STACK
      DB 1024 DUP (" ")
PILA ENDS
;
; ********************************************************************************************
;
; A continuación definimos el segmento de código.
;
; ********************************************************************************************
;
CODIGO SEGMENT
      ASSUME CS:CODIGO, SS:PILA, DS:DATOS
;
; ********************************************************************************************
;
; A continuación definimos el procedimiento principal que llama a los siguientes procedimientos:
;           LeerCadena:       Solicita y lee por teclado una cadena máxima de 80 caracteres.
;           TratarCadena:     Comprime los dígitos leídos.
;           Enter:            Produce un salto de línea por pantalla.
;           ImprimirCadena:   Escribe la frase codificada por pantalla.
; Parámetros que pasa:        NINGUNO.
; Parámetros que recibe:      NINGUNO.
;
; ********************************************************************************************
;
Principal PROC FAR
      MOV AX, DATOS
      MOV DS, AX
      LEA DX, MsgPedirFrase      ; Ponemos en DS:DX la dirección de la cadena que vamos a imprimir.
      CALL ImprimirCadena        ; imprimimos la cadena.
      LEA DX, Frase              ; Ponemos en DS:DX la dirección en la que dejar la frase leída.
      CALL LeerCadena            ; Leemos la cadena.
      CALL TratarCadena          ; Comprimimos la cadena.
      CALL Enter                 ; Realizamos un salto de línea.
      LEA DX, MsgResultado       ; Ponemos en DS:DX la dirección de la cadena que vamos a imprimir.
      CALL ImprimirCadena        ; Se imprime la cadena.
      LEA DX, FraseComprimida    ; Ponemos en DS:DX la cadena comprimida.
      CALL ImprimirCadena        ; Imrpimimos la cadena comprimida.
      MOV AX, TerminarPrg        ; Terminamos el programa con la función 4C00h de la INT 21h
      INT 21h
Principal 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 imrpimir en DS:DX.
;
; ********************************************************************************************
;
ImprimirCadena PROC NEAR
      PUSH AX                  ; Guardamos el registro que vamos a utilizar.
      MOV AH, EscribeCadena    ; Imprimimos la cadena mediante la función 09h
      INT 21h                  ; de la INT 21h.
      POP AX                   ; Recuperamos el contenido que tenía el registro antes de llamar
                               ; al procedimiento.
      RET
ImprimirCadena ENDP
;
; ********************************************************************************************
;
; PROCEDIMIENTO:     LeerCadena
; OBJETIVOS:         Leer una cadena de 80 caracteres como máximo Deja la cadena leída en
;                    la estructura Frase. Emplea la función 0Ah de la INT 21h.
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  La dirección de la cadena a leer en DS:DX.
;
; ********************************************************************************************
;
LeerCadena PROC NEAR
      PUSH AX                  ; Guardamos el registro que vamos a utilizar.
      MOV AH, LeeCadena        ; Leemos la cadena mediante la función 09h
      INT 21h                  ; de la INT 21h.
      POP AX                   ; Recuperamos el contenido que tenía el registro antes de llamar
                               ; al procedimiento.
      RET
LeerCadena ENDP
;
; ********************************************************************************************
;
; PROCEDIMIENTO:     TratarCadena
; OBJETIVOS:         Comprime la cadena de carácteres almacenada en Frase y deja el resultado en
;                    FraseComprimida. Lo que se hace realmente es análogo al  BCD empaquetado.
; Llama a los procedimientos:
;          Convertir:      Convierte al valor númerico cada uno de los dígitos leídos.
;          JuntarEnByte:   Junta cada dos dígitos en un único byte.
; Parámetros que pasa:    Dirección de las cadenas Frase y FraseComprimida.
; Parámetros que recibe:  NINGUNO.
;
; ********************************************************************************************
;
```

```
TratarCadena PROC NEAR
        PUSH BX             ; Guardamos los registros que vamos a emplear.
        PUSH SI
        LEA BX, Frase       ; Ponemos en DS:BX la dirección de memoria en la que comienza la
        CALL Convertir      ; cadena leída y llamamos al procedimento Convertir.
        LEA BX, Frase       ; Ponemos en DS:BX la dirección de memoria en la que comienza la cadena
                            ; leída.
        LEA SI, FraseComprimida     ; Ponemos en DS:SI la dirección de memoria en la que comienza la
                                    ; cadena comprimida.
        CALL JuntarEnByte           ; Unimos las dos cifras hexadecimales en una sola.
        POP SI                      ; Recuperamos el contenido que tenían los registros antes
        POP BX                      ; de llamar al procedimiento.
        RET
TratarCadena ENDP
;
; ********************************************************************************************
;
; PROCEDIMIENTO:    Convertir.
; OBJETIVOS:        Convierte los dígitos de su código ASCII a su valor numérico
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  En BX la dirección de la cadena a convertir.
;
; ********************************************************************************************
;
Convertir PROC NEAR
        PUSH CX             ; Guardamos el contenido del registro que vamos a emplear.
        INC BX
        XOR CX, CX
        MOV CL, [BX]        ; Ponemos en CX el número de caracteres leído.
BucleConvertir:
        INC BX
        MOV AL, [BX]
        SUB AL, CorregirNumero      ; Convertimos de Código ASCII al valor numérico.
        MOV [BX], AL
        LOOP BucleConvertir
        POP CX              ; Recuperamos el contenido que tenía antes el registro.
        RET
Convertir ENDP
;
; ********************************************************************************************
;
; PROCEDIMIENTO:    JuntarEnByte
; OBJETIVOS:        Lo que se hace realmente es análogo al  BCD empaquetado.
; Llama al procedimiento:
;         JuntarDos:      Junta cada dos dígitos en un único byte.
; Parámetros que pasa:    Dirección de las cadenas Frase y FraseComprimida.
; Parámetros que recibe:  Dirección de las cadenas Frase y FraseComprimida.
;
; ********************************************************************************************
;
JuntarEnByte PROC NEAR
        PUSH CX             ; Guardamos el contenido del registro que vamos a emplear.
        INC BX
        XOR CX, CX
        MOV CL, [BX]        ; Vemos cuantos caracteres hemos leído.
        TEST CL, 01h        ; Si es un número impar lo tratamos adecuadamente-
        JNZ EsImpar
        SHR CL, 1           ; Desplazamos el número de valores una posición a la derecha, lo que
        JMP BucleJuntar     ; es equivalente a dividir por dos.
```

```
EsImpar:
        SHR CL, 1               ; Dividimo por dos despalazando el registro a la derecha y lo
        INC CL                  ; incrementamos en uno.
BucleJuntar:
        CALL JuntarDos          ; Llamamos al procedimiento que junta las dos cifras
        LOOP BucleJuntar
        MOV CL, '$'             ; Colocamos el '$' para indicar el final de la cadena
        MOV [SI], CL
        POP CX                  ; Recuperamos el contenido que tenía antes el registro.
        RET
JuntarEnByte ENDP
;
; ********************************************************************************
;
; PROCEDIMIENTO:    JuntarDos
; OBJETIVOS:        Junta dos caracteres en un único byte.
;
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  Dirección de las cadenas Frase y FraseComprimida.
;
; ********************************************************************************
;
JuntarDos PROC NEAR
        PUSH DX                 ; Guardamos el contenido de los registros que vamos a utilizar.
        PUSH CX
        INC BX
        MOV DH, [BX]            ; Accedemos al primer, tercer, quinto, ... dígito de la cadena.
        INC BX
        MOV DL, [BX]            ; Accedemos al segundo, cuarto, sexto, ...  dígito de la cadena
        MOV CX, 0004h
        SHL DH, CL              ; Desplazamos el dígito impar cuatro posiciones a la izquierda.
        ADD DH, DL              ; Sumaos los dos dígitos.
        MOV [SI], DH            ; Dejamos el resultado en la posición correspondiente a
        INC SI                  ; la FraseComprimida-
        POP CX                  ; Recuperamos el contenido de los registros que hemos empleado.
        POP DX
        RET
JuntarDos ENDP
;
; ********************************************************************************
;
; PROCEDIMIENTO:    Enter
; OBJETIVOS:        Realiza un salto de línea por pantalla.
; Llama al procedimiento:
;       ImprimirCadena:     Muestra una cadena de caracteres terminada en $ mediante la
;                           función 09h de la INT 21h
;
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  NINGUNO.
;
; ********************************************************************************
;
Enter PROC NEAR
        PUSH DX                 ; Guardamos el contenido del registro.
        LEA DX, MsgEnter        ; Ponemos en DS:DX la dirección de la cadena ENTER
        CALL ImprimirCadena     ; Llamamos a imprimir la cadena.
        POP DX                  ; Recuperamos el contenido del registro.
        RET
Enter ENDP
```

```
CODIGO ENDS
END Principal
```

5º) Realizar un programa en ensamblador que lea dos cadenas de caracteres y escriba como salida un carácter de cada cadena de forma alternada hasta que se termine la cadena más corta y a continuación el resto de la cadena más larga

Ejemplos

C:\> JUNTA

Introduzca la primera frase, por favor
hola pepe
Introduzca la segunda frase, por favor
hola juan
Las frases mezcladas serán:
hhoollaa pjeupaen

C:\> JUNTA

Introduzca la primera frase, por favor
Hola pepe
Introduzca la segunda frase, por favor
Hola Juan, ¿como estás?
Las frases mezcladas serán:
HHoollaa pJeupaen, ¿como estás?

SOLUCIÓN:

```
; ********************************************************************************************
;
; A continuación definimos el segmento de datos.
;
; ********************************************************************************************
;
DATOS SEGMENT
    TerminarPrg         EQU   4C00h          ; Función de la INT 21h
    CorregirNumero      EQU   30h
    EscribeCadena       EQU   09h            ; Función de la INT 21h
    EscribeCaracter     EQU   02h            ; Función de la INT 21h
    LeeCadena           EQU   0Ah            ; Función de la INT 21h
    Espacio             EQU   20h
    MsgPedirFrase1       DB   'Introduzca la primera frase, por favor', 10, 13, '$'
    MsgPedirFrase2       DB   'Introduzca la segunda frase, por favor', 10, 13, '$'
    MsgResultado         DB   'Las frases mezcladas serán: ', 10, 13, '$'
    MsgEnter             DB   10, 13, '$'; Cadena de texto que simula un ENTER
    Frase1               DB   81, 0      ; Estructura para almacenar la primera frase.
                         DB   81 DUP (0)
    Frase2               DB   81, 0      ; Estructura para almacenar la primera frase.
                         DB   81 DUP (0)
    Menor                DB   00h        ; Menor número de caracteres.
    Mayor                DB   00h        ; Menor número de caracteres.
    DirMayor             DW   0000h      ; Dirección de la cadena más larga.
DATOS ENDS
;
; ********************************************************************************************
;
; A continuación definimos el segmento de pila.
;
; ********************************************************************************************
;
PILA SEGMENT STACK
    DB 1024 DUP (" ")
PILA ENDS
```

```
;
; ********************************************************************************************
;
; A continuación definimos el segmento de código.
;
; ********************************************************************************************
;
CODIGO SEGMENT
      ASSUME CS:CODIGO, SS:PILA, DS:DATOS
;
; ********************************************************************************************
;
; A continuación definimos el procedimiento principal que llama a los siguientes procedimientos:
;           LeerCadena:       Solicita y lee por teclado una cadena máxima de 80 caracteres.
;           PedirFrases:      Solicita  lee dos frases por teclado.
;           Enter:            Produce un salto de línea por pantalla.
;           ImprimirCadena: Escribe la frase codificada por pantalla.
;           ImprimirFrases:   Escrbe las dos frases juntas.
; Parámetros que pasa:      NINGUNO.
; Parámetros que recibe:    NINGUNO.
;
; ********************************************************************************************
;
Principal PROC FAR
      MOV AX, DATOS
      MOV DS, AX
      CALL PedirFrases          ; Solicitamos y leemos dos frases por pantalla.
      CALL Enter                ; Realizamos un salto de línea.
      LEA DX, MsgResultado      ; Ponemos en DS:DX el mensaje resultado.
      CALL ImprimirCadena       ; Escribimos el mensaje del resultado.
      CALL Enter                ; Realizamos un salto de línea.
      CALL ImprimirFrases       ; Imprimimos las dos frases juntas.
      MOV AX, TerminarPrg       ; Terminamos el programa con la función 4c00h de
      INT 21h                   ; la INT 21h.
Principal ENDP
```

```
;
; ********************************************************************************************
;
; PROCEDIMIENTO:     PedirFrases
; OBJETIVOS:         Realiza un salto de línea por pantalla.
;
; A continuación definimos el procedimiento principal que llama a los siguientes procedimientos:
;          LeerCadena:     Solicita y lee por teclado una cadena máxima de 80 caracteres.
;          Enter:          Produce un salto de línea por pantalla.
;          ImprimirCadena: Escribe una cadena de caracteres por pantalla.
;
; Parámetros que pasa:     La dirección en DS:DX de las cadenas MsgPedirFrase1, Frase1,
;                          MsgPedirFrase2, Frase2
; Parámetros que recibe:   NINGUNO.
; ********************************************************************************************
;
PedirFrases PROC NEAR
      PUSH DX                    ; Guardamos el registro que vamos a utilizar.
      LEA DX, MsgPedirFrase1     ; Ponemos en DS:DX la dirección de la cadena MsgPedirFrase1.
      CALL ImprimirCadena        ; Imprimimos la cadena.
      LEA DX, Frase1             ; Ponemos en DS:DX la dirección de la cadena Frase1.
      CALL LeerCadena            ; Leemos la cadena.
      CALL Enter                 ; Ralizamos un salto de línea.
      LEA DX, MsgPedirFrase2     ; Ponemos en DS:DX la dirección de la cadena MsgPedirFrase2.
      CALL ImprimirCadena        ; Imprimimos la cadena.
      LEA DX, Frase2             ; Ponemos en DS:DX la dirección de la cadena Frase2.
      CALL LeerCadena            ; Leemos la cadena.
      POP DX                     ; Recuperamos el contenido del registro que hemos utilizado.
      RET
PedirFrases 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 imrpimir en DS:DX.
;
; ********************************************************************************************
;
ImprimirCadena PROC NEAR
      PUSH AX                    ; Guardamos el registro que vamos a utilizar.
      MOV AH, EscribeCadena      ; Imprimimos la cadena mediante la función 09h
      INT 21h                    ; de la INT 21h.
      POP AX                     ; Recuperamos el contenido que tenía el registro antes de llamar
                                 ; al procedimiento.
      RET
ImprimirCadena ENDP
```

```
;
; ********************************************************************************************
;
; PROCEDIMIENTO:     LeerCadena
; OBJETIVOS:         Leer una cadena de 80 caracteres como máximo Deja la cadena leída en
;                    la estructura Frase. Emplea la función 0Ah de la INT 21h.
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  La dirección de la cadena a leer en DS:DX.
;
; ********************************************************************************************
;
LeerCadena PROC NEAR
     PUSH AX                ; Guardamos el registro que vamos a utilizar.
     MOV AH, LeeCadena      ; Leemos la cadena mediante la función 09h
     INT 21h                ; de la INT 21h.
     POP AX                 ; Recuperamos el contenido que tenía el registro antes de llamar
                            ; al procedimiento.
     RET
LeerCadena ENDP
;
; ********************************************************************************************
;
; PROCEDIMIENTO:     ImprimirFrases
; OBJETIVOS:         Imprime las frases una a continuación de la otra prolongando la más larga
;                    al final de la alternancia de las dos cadenas.
;
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  NINGUNO.
;
; ********************************************************************************************
;
ImprimirFrases PROC NEAR
     PUSH SI                ; Guardamos el contenido de los registros que vamos a emplear.
     PUSH BX
     PUSH CX
     PUSH AX
     PUSH DX
     XOR CX, CX             ; Ponemos el registro contador a cero
     LEA BX, Frase1         ; Ponemos en DS:BX la dirección de la Frase1.
     LEA SI, Frase2         ; Ponemos en DS:SI la dirección de la Frase2.
     INC BX
     INC SI
     MOV DH, [BX]           ; Accedemos al número de caracteres leídos en Frase1.
     MOV DL, [SI]           ; Accedemos al número de caracteres leídos en Frase2.
     CMP DH, DL             ; Comparamos para ver que cadena es la mayor.
     JA MayorDH
     MOV CL, DH             ; Ponemos en el contador el número menor
     MOV Mayor, DL          ; Dejamos en Mayor el número de caracteres de la cadena mayor.
     MOV DirMayor, SI       ; Ponemos en DirMayor la dirección de la cadena mayor.
     JMP Seguir
MayorDH:
     MOV DirMayor, BX       ; Ponemos en DirMayor la dirección de la cadena mayor.
     MOV CL, DL             ; Ponemos en el contador el número menor
     MOV Mayor, DH          ; Dejamos en Mayor el número de caracteres de la cadena mayor.
Seguir:
     MOV Menor, CL          ; Ponemos en menor la longitud de la cadena menor.
     MOV AH, EscribeCaracter ; Ponemos en AH la función 02h de la INT 21h para eacribir
                            ; caracteres de uno en uno estando su código ASCII en DL.
BucleEscribir:
```

```
        INC SI                  ; Incrementamos al primer caracter de la cadena de cada frase
        INC BX
        MOV DL, [BX]            ; Llevamos un carácter de Frase1 a DL
        INT 21h                 ; y lo imprimimos.
        MOV DL, [SI]            ; Llevamos un carácter de Frase2 a DL
        INT 21h                 ; y lo imprimimos.
        LOOP BucleEscribir      ; Repetimos tantas veces como tengamos en el contador.
        MOV DH, Mayor           ; Recuperamos los tamaños de las dos cadenas
        MOV DL, Menor
        SUB DH, DL              ; Calculamos la diferencia de las dos cadenas.
        JZ FinalImprimir        ; Si son iguales hemos terminado
        XOR CX, CX
        MOV CL, DH              ; Ponemos la diferencia calcuala en el registro contador CX.
        XOR DX, DX              ; Ponemos en DX el tamaño de la cadenamenor.-
        MOV DL, Menor
        MOV BX, DirMayor        ; Ponemos en BX la dirección de la cadena mayor
        ADD BX, DX              ; Incrementamos en DX la posición en la cadena
BucleResto:
        INC BX                  ; Vamos incrementando e imprimiendo los caracteres que quedan
        MOV DL, [BX]
        INT 21h
        LOOP BucleResto
FinalImprimir:
        POP DX                  ; Recuperamos el contenido de los registros que hemos utilizado.
        POP AX
        POP CX
        POP BX
        POP SI
        RET
ImprimirFrases ENDP
; ********************************************************************************************
;
; PROCEDIMIENTO:    Enter
; OBJETIVOS:        Realiza un salto de línea por pantalla.
; Llama al procedimiento:
;         ImprimirCadena:       Muestra una cadena de caracteres terminada en $ mediante la
;                               función 09h de la INT 21h
; Parámetros que pasa:  NINGUNO.
; Parámetros que recibe:  NINGUNO.
;
; ********************************************************************************************
;
Enter PROC NEAR
        PUSH DX                 ; Guardamos el contenido del registro.
        LEA DX, MsgEnter        ; Ponemos en DS:DX la dirección de la cadena ENTER
        CALL ImprimirCadena     ; Llamamos a imprimir la cadena.
        POP DX                  ; Recuperamos el contenido del registro.
        RET
Enter ENDP
CODIGO ENDS
END Principal
```

6º) Realizar un programa en ensamblador que lea un carácter por teclado e indique si el carácter leído es una letra o un número. En el caso de que sea una letra deberá indicar el número de orden que ocupa dicha letra en el alfabeto. El resultado no es necesario pasarlo a decimal, vale indicar la posición con el valor hexadecimal.

Ejemplos:

C:\>DECIDE

Introduzca una letra o un número, por favor

1

El carácter introducido es un número

C:\>DECIDE

Introduzca una letra o un número, por favor

p

El carácter introducido es un letra y ocupa en el alfabeto la posición: 10h

SOLUCIÓN:

```
; ********************************************************************************************
;
; A continuación definimos el segmento de datos.
;
; ********************************************************************************************
;
DATOS SEGMENT
      TerminarPrg     EQU     4C00h
      CorregirNumero  EQU     30h
      CorregirLetra   EQU     07h
      EscribeCadena   EQU     09h
      EscribeCaracter EQU     02h
      LeeCaracter     EQU     01h
      Espacio         EQU     20h
      CERO            EQU     00h
      HACHE           EQU     68h
      MsgPedirCaracter DB     'Introduzca una letra o un número, por favor', 10, 13, '$'
      MsgResNum        DB     'El carácter introducido es un número', 10, 13, '$'
      MsgResLetra      DB     'El carácter introducido es un letra$'
      MsgPosición      DB     ' y ocupa en el alfabeto la posición: $'
      MsgEnter         DB     10, 13, '$'          ; Simulación de un enter.
      Caracter         DB     00h
      NumeroMayor EQU     09h             ; Valor del número mayor.
      DigitoMayor     EQU     39h             ; Código ASCII del código numérico mayor.
      DigitoMenor     EQU     30h             ; Código ASCII del código numérico menor.
      AMinuscula      EQU     20h             ; Valor a sumar para pasar una letra a minuscula.
      ZMayuscula      EQU     5Ah             ; Código ASCII del código de la Z.
      AMayuscula      EQU     41h             ; Código ASCII del código de la A.
DATOS ENDS
```

```
; ********************************************************************************************
;
; A continuación definimos el segmento de pila.
;
; ********************************************************************************************
;
PILA SEGMENT STACK
      DB 1024 DUP (" ")
PILA ENDS
;
; ********************************************************************************************
;
; A continuación definimos el segmento de código.
;
; ********************************************************************************************
;
CODIGO SEGMENT
      ASSUME CS:CODIGO, SS:PILA, DS:DATOS
;
; ********************************************************************************************
;
; A continuación definimos el procedimiento principal que llama a los siguientes procedimientos:
;           PedirCaracter:    Solicita y lee por teclado un carácter.
;           Enter:            Produce un salto de línea por pantalla.
;           DecidirSalida:    Escribe la salida por pantalla.
;
; Parámetros que pasa:      NINGUNO.
; Parámetros que recibe:    NINGUNO.
;
; ********************************************************************************************
;
Principal PROC FAR
      MOV AX, DATOS             ; Inicializamos el segmento de datos.
      MOV DS, AX
      CALL PedirCaracter        ; Solicitamos y leemos un carácter por teclado.
      CALL Enter                ; Escribimos un salto de línea por pantalla.
      CALL DecidirSalida        ; Evaluamos e imprimimos el resultado por pantalla.
      MOV AX, TerminarPrg       ; Indicamos al DOS que hemos terminado la ejecución
      INT 21h                   ; del programa.
Principal ENDP
;
; ********************************************************************************************
;
; PROCEDIMIENTO:      PedirCaracter
; OBJETIVOS:          Solicita y lee un carácter del teclado mediante la función 01h de la INT 21h.
; Llama al procedimiento:
;           ImprimirCadena:         Escribe una cadena de caracteres terminada en $ por pantalla.
; Parámetros que pasa:      La  dirección de la cadena a imrpimir en DS:DX.
; Parámetros que recibe:    NINGUNO.
********************************************************************************************
```

```
;
PedirCaracter PROC NEAR
      PUSH DX                    ; Guardamos el contenido de los registros que vamos a utilizar.
      PUSH AX
      LEA DX, MsgPedirCaracter   ; Cargamos en DS:DX la dirección de la cadena a imprimir.
      CALL ImprimirCadena        ; Imprimimos la cadena.
      MOV AH, LeeCaracter        ; Leemos un caracter mediante la funcion 01 de la INT 21k
NINGUNOh.
      INT 21h
      MOV Caracter, AL           ; Llevamos el código ASCII del carácter leído a la variable Caracter.
      POP AX                     ; Recuperamos el contenido de los registros que tenían antes de
      POP DX                     ; llamar al procedimiento
      RET
PedirCaracter 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 imrpimir en DS:DX.
;
; ********************************************************************************************
;
ImprimirCadena PROC NEAR
      PUSH AX
      MOV AH, EscribeCadena
      INT 21h
      POP AX
      RET
ImprimirCadena ENDP
;
; ********************************************************************************************
;
; PROCEDIMIENTO:     Enter
; OBJETIVOS:         Realiza un salto de línea por pantalla.
; Llama al procedimiento:
;          ImprimirCadena:        Muestra una cadena de caracteres terminada en $ mediante la
;                                 función 09h de la INT 21h
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  NINGUNO.
;
; ********************************************************************************************
;
Enter PROC NEAR
      PUSH DX                    ; Guardamos el contenido del registro.
      LEA DX, MsgEnter           ; Ponemos en DS:DX la dirección de la cadena ENTER
      CALL ImprimirCadena        ; Llamamos a imprimir la cadena.
      POP DX                     ; Recuperamos el contenido del registro.
      RET
Enter ENDP
```

```
; ********************************************************************************************
;
; PROCEDIMIENTO:     DecidirSalida
; OBJETIVOS:         Evalua e imprime la salida del programa
; Llama a los procedimientos:
;         CalcularOrden:          Calcula el número de orden de la letra dentro del alfabeto.
;         CalcularASCII:          Calcula el código ASCII del elemento a imprimir.
;         ImprimirCadena:         Muestra una cadena de caracteres terminada en $ mediante la
;                                 función 09h de la INT 21h
;
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  NINGUNO.
;
; ********************************************************************************************
;
DecidirSalida PROC NEAR
      PUSH AX                   ; Guardamos el contenido de los registros que vamos a emplear.
      PUSH CX
      PUSH DX
      MOV AL, Caracter          ; Llevamoa a AL el código ASCII del carácter leído.
      CMP AL, DigitoMayor       ; Comparamos AL con para ver si es ≤ 9
      JBE SeguirNumero          ; Si es así se trata de un núemro
EsLetra:
      CALL CalcularOrden        ; En otro caso calculamos el número de orden de AL
      LEA DX, MsgResLetra       ; Ponemos en DS:DX la dirección de MsgLEtra
      CALL ImprimirCadena       ; Imprimimos la cadena.
      LEA DX, MsgPosición       ; Ponemos en DS:DX la dirección de MsgPosición.
      CALL ImprimirCadena       ; Imprimimos la cadena.
      CALL CalcularAscci        ; Calculamos el código ASCII de la posición.
      MOV CX, AX                ; Llevamos a CX el contenido de AX.
      MOV AH, EscribeCaracter   ; Ponemos en AH la función 02
      CMP CH, CERO              ; comparamos CH con Cero
      JE SegundaCifra           ; Si es cero solamente imprimo la segunda cifra
      MOV DL, CH                ; Llevamos el código ASCII de la primera cifra a DL,
      INT 21h                   ; y lo imprimimos mediante la función 02 de la INT 21h
SegundaCifra:
      MOV DL, CL                ; Llevamos el código ASCII de la segunda cifra a DL,
      INT 21h                   ; y lo imprimimos mediante la función 02 de la INT 21h
      MOV DL, HACHE             ; Escribimos la H para indicar que el valor es hexadecimal.
      INT 21h
      JMP Final                 ; Hemos terminado.
SeguirNumero:
      CMP AL, DigitoMenor       ; Comparamos con el dígito menor.
      JAE EsNumero              ; Si es ≥ 0 es un número.
      JMP EsLetra               ; Si no será una letra.
EsNumero:
      LEA DX, MsgResNum         ; Ponemos en DS:DX la dirección de MsgResNum
      CALL ImprimirCadena       ; Imprimimos la cadena.
Final:
      POP DX                    ; Recuperamos el contenido que tenían los registros antes de
      POP CX                    ; llamar al procedimiento.
      POP AX
      RET
DecidirSalida ENDP
```

```
; ********************************************************************************
;
; PROCEDIMIENTO:     CalcularOrden
; OBJETIVOS:         Calcula el número de orden del carácter alfabético leído.
;                    Devuelve en AL el número de orden.
;
; Parámetros que pasa:    NINGUNO
; Parámetros que recibe:  En AL el código ASCII del carácter leído.
;
; ********************************************************************************
;
CalcularOrden PROC NEAR
      CMP AL, ZMayuscula        ; Comparamos con la Z mayúscula.
      JBE SeguirOrden           ; Si es mayúscula calculamos el orden.
      SUB AL, AMinuscula        ; En otro caso pasamos a mayuscula.
SeguirOrden:
      SUB AL, AMayuscula        ; Restamos el código ASCII de la A mayuscula.
      INC AL                    ; Incrementamos en 1.
      RET
CalcularOrden ENDP
; ********************************************************************************
;
; PROCEDIMIENTO:     CalcularAscii
; OBJETIVOS:         Realiza la conversión a código ASCII del número leído.
;                    Devuelve en AX el código ASCII del número de orden.
;
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  En AL el número de orden del carácter.
;
; ********************************************************************************
;
CalcularAscci PROC NEAR
      PUSH CX                   ; Guardamos el contenido de CX.
      XOR AH, AH                ; Ponemos AH a cero.
      CMP AL, NumeroMayor       ; Comparamos AL con 9
      JBE SumaYSigue            ; SI es ≤ 9 Seguimos
      MOV AH, AL                ; En otro caso copiamos en AH el valor de AL.
      MOV CX, 0004h             ; Ponemos en CX 4.
      SHR AH, CL                ; Nos quedamos con la primera cifra en AH
      AND AL, 0Fh               ; Nos quedamos con la segunda cifra en AL
      CMP AH, NumeroMayor       ; Comparamos con el número mayor.
      JBE CorrigeNumero         ; Si es ≤ 9 corregimos el número.
      ADD AH, CorregirLetra     ; Sumamos la correción para número entre A y F.
CorrigeNumero:
      ADD AH, CorregirNumero    ; Sumamos la correccón para número entre 0 y 9.
SumaYSigue:
      CMP AL, NumeroMayor       ; Comparamos con el número mayor.
      JBE CorrNum2              ; Si es ≤ 9 corregimos el número.
      ADD AL, CorregirLetra     ; Sumamos la correción para número entre A y F.
CorrNum2:
      ADD AL, CorregirNumero    ; Sumamos la correccón para número entre 0 y 9.
      POP CX                    ; Recuperamos el contenido de CX.
      RET
CalcularAscci ENDP
CODIGO ENDS
END Principal
```