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

(I.T.I. GESTIÓN - I.T.I. SISTEMAS)

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 una Extensión de un fichero, deberá sacar el listado de todos aquellos ficheros que no tengan esa extensión.
c) La salida tendrá que estar ordenada por nombre y además si son más de 25 ficheros deberá presentarse de pantalla en pantalla.

Ejemplos:

C:\>INVDIR

El comando INVDIR saca todos los ficheros que no tengan la extensión especificada, ordenados por nombre y deteniéndose la salida de pantalla en pantalla..

La sintaxis es:

"INVDIR [/? | extensión]"

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 SACAMOS UN DIRECTORIO ORDENADO POR NOMBRE Y MEDIANTE UN “PIPE”
REM REDIRIGIMOS LA SALIDA AL COMANDO FIND EN EL QUE SE MOSTRARÁN LAS
REM LÍNEAS QUE NO CONTENGAN DETERMINADA CADENA. POR ÚLTIMO, TAMBIÉN
REM REDIRIGIMOS ESTA SALIDA, PERO ESTA VEZ AL COMANDO “MORE” PARA  PODER
REM VISUALIZAR LA SALIDA CADA 25 LÍNEAS.
DIR /B /ON | FIND /I /V "%1" | MORE

REM HEMOS TERMINADO Y SALTAMOS A FIN.
GOTO Fin

REM MOSTRAMOS LA AYUDA DEL COMANDO.
:Ayuda
  ECHO.
  ECHO El comando INVDIR saca todos los ficheros que no
  ECHO tengan la extensión especificada, ordenados por nombre
  ECHO y deteniéndose la salida de pantalla en pantalla.
  ECHO.
  ECHO La sintaxis es:
  ECHO.
  ECHO "INVDIR [/? | extensión]"
:Fin
```

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

a) Añada a la variable PATH del fichero AUTOEXCEC.BAT los nombres de todos los directorios que vengan a continuación.
b) No se añadirán aquellos directorios que ya estén en la variable PATH.
c) Si el directorio especificado no existe no se añadirá a la variable PATH ni tampoco se creará en el disco duro.
d) El proceso seguirá hasta que no existan más parámetros.

Ejemplos:

C:\>ADD2BAT

El comando ADD2BAT añade a la variable PATH del fichero AUTOEXEC.BAT todos los directorios que vienen a continuación, Siempre que no estén ya y que el directorio exista en el disco duro.

La sintaxis es:

"ADD2BAT [/?| Directorio1 [Directorio2 [...]]]"

C:\>ADD2BAT c:\tmp c:\dos c:\usr\pepe

SOLUCIÓN:

```
REM DESACTIVAMOS 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 BUSCAMOS LA CADENA “PATH” EN EL FICHERO AUTOEXEC.BAT Y REDIRIGIMOS
REM LA SALIDA A UN FICHERO TEMPORAL
FIND /I "PATH" AUTOEXEC.BAT > Temporal.tmp

REM RECORREMOS TODOS LOS PARÁMETROS MEDIANTE UN BUCLE.
:BuclePath

REM BUSCAMOS EL PARÁMETRO EN EL FICHERO TEMPORAL.
  FIND /I ";%1" Temporal.tmp >NUL

REM SI NO SE ENCUENTRA LO AÑADIMOS.
  IF ERRORLEVEL 1 GOTO Anadir

REM EN OTRO CASO SALTAMOS AL FINAL DEL BUCLE.
  GOTO FinBucle

REM VERIFICAMOS QUE EXISTA EL DIRECTORIO ANTES DE AÑADIRLO A LA
REM VARIABLE TEMPORAL “TMP”
:Anadir
  IF EXIST %1\NUL SET TMP=%TMP%;%1

REM ACTUALIZAMOS EL CONTADOR DEL BUCLE
:FinBucle
  SHIFT

REM SI NO HAY MÁS PARÁMETROS SALTAMOS A “TratarPath”
  IF "%1"=="" GOTO TratarPath
  GOTO BuclePath
```

```
REM SACAMOS LA AYUDA DEL COMANDO
:Ayuda
  ECHO.
  ECHO El comando ADD2BAT añade al PATH del fichero AUTOEXEC.BAT
  ECHO todos los directorios que vengan a continuación. Siempre
  ECHO que no estén ya. Si el directorio no existe, no se añadirá.
  ECHO.
  ECHO La sintaxis es:
  ECHO.
  ECHO "ADD2BAT [/?| Directorio1 [Directorio2 [ ... ] ] ]"
  GOTO Fin

REM VERIFICAMOS QUE LA VARIABLE “TMP” TENGA ALGO.
:TratarPath
  IF "%TMP%"=="" GOTO Fin

REM AÑADIMOS COMO LÍNEA LA AUTOEXEC.BAT LA LÍNEA SIGUIENTE
  ECHO SET PATH=%PATH%%TMP% >> AUTOEXEC.BAT
  DEL Temporal.tmp
  SET TMP=
:Fin
```

3º) Se desea realizar un fichero BAT que elimine todas las líneas de un fichero que se pasa como parámetro que contengan una palabra que se pasa también como parámetro.

Ejemplos:

C:\> FILTRA

El comando FILTRA elimina todas las líneas del fichero que contenga una determinada palabra.

La sintaxis es:

"FILTRA [/? | palabra fichero]"

C:\> FILTRA pedro c:\tmp\prueba.txt

SOLUCIÓN:

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


REM CONTROLAMOS QUE NO SE INTRODUZCAN PARÁMETROS O QUE SE INTRODUZCA
EL
REM PARÁMETRO /? O SI NO HAY DOS PARÁMETROS EN CUYO CASO SACAMOS LA
REM AYUDA DEL COMANDO
IF "%1"=="" GOTO Ayuda
IF "%1"=="/?" GOTO Ayuda
IF "%2"=="" GOTO Ayuda

REM SI NO EXISTE EL FICHERO SE INDICA CON UN MENSAJE DE ERROR
IF NOT EXIST %2 GOTO Error

REM SACAMOS EL CONTENIDO DEL FICHERO Y GUARDAMOS LAS LÍNEAS QUE NO
REM CONTENGAN EL  TEXTO EN UN FICHERO TEMPORAL.
TYPE %2 | FIND /I /V "%1" > Temporal.Tmp

REM COPIAMOS EL FICHERO TEMPORAL SOBRE LE INICIAL
COPY Temporal.Tmp %2 /Y >NUL
```

```
REM ELIMINAMOS EL FICHERO TEMPORAL.
DEL Temporal.Tmp
GOTO Fin

REM VISUALIZAMOS EL FICHERO TEMPORAL.
:Error
  ECHO EL fichero %2 no existe.
  GOTO Fin

REM VISUALIZAMOS LA AYUDA DEL COMANDO
:Ayuda
  ECHO.
  ECHO El comando FILTRA elimina todas las líneas de un único fichero
  ECHO que contengan una determinada palabra.
  ECHO.
  ECHO La sintaxis es:
  ECHO.
  ECHO "FILTRA [/? | palabra fichero]"
  GOTO Fin
:Fin
```

4º) Realizar un programa en ensamblador que lea una cadena máxima de 80 caracteres y los imprima de forma ordenada. Como método de ordenación se recomienda emplear el algoritmo de la burbuja.

Ejemplos:

C:\> ORDENA

Introduzca una cadena de un máximo de 80 caracteres, por favor
La jaca Paca está atada a la estaca.
Los caracteres ordenados alfabéticamente son:
.LPaaaaaaaaaaaacccdeejlsstttá

C:\> ORDENA

Introduzca una cadena de números entre 0 y 9 por favor
9876543210
Los caracteres ordenados alfabéticamente son:
0123456789

SOLUCIÓN:

```
; ********************************************************************************************
;
; A continuación definimos el segmento de datos.
;
; ********************************************************************************************
;

DATOS SEGMENT
    TerminarPrg     EQU   4C00h          ; Función de la INT 21h
    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            ; Código ASCII del espacio en blanco.
    MsgPedirFrase   DB    'Introduzca una frase (máximo de 80 caracteres), por favor', 10, 13, '$'
    MsgResultado    DB    'Los caracteres ordenados alfabéticamente son: ', 10, 13, '$'
    MsgEnter        DB    10, 13, '$'  ; Cadena de caracteres que simula un ENTER.
    Frase           DB    81, 0        ; Estructura para poder leer una cadena de 80
                    DB    81 DUP (0)
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:
;		PedirFrase:	Solicita y lee por teclado una cadena máxima de 80 caracteres.
;		Burbuja:	Ordena la cadena alfabéticamente.
;		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
	CALL PedirFrases		; Pedimos la cadena de caracteres.
	CALL Enter			; Provocamos un salto de línea por pantalla.
	LEA DX, MsgResultado		; Ponemos en DS:DX la dirección de la cadena que vamos a imprimir.
	CALL ImprimirCadena		; Imprimimos la cadena.
	CALL Enter			; Provocamos un salto de línea por pantalla.
	CALL Burbuja			; Ordenamos la cadena por el método de la burbuja.
	LEA DX, Frase			; Ponemos en DS:DX la dirección de la cadena que vamos a imprimir.
	ADD DX, 2h			; Saltamos las dos primeras posiciones de la cadena.
	CALL ImprimirCadena		; Imprimimos la cadena.
	MOV AX, TerminarPrg		; Terminamos el programa con la función 4C00h de la INT 21h
	INT 21h
Principal ENDP
```

```
;
; ********************************************************************************************
;
; PROCEDIMIENTO:     PedirFrases.
; 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.
; Llama a los procedimientos:
;          ImprimirCadena:        Imprime una cadena de caracteres terminada en $.
           LeerCadena:            Lee una cadena de caracteres.
           Enter:                 Produce un salto de línea por pantalla.
; Parámetros que pasa:     NINGUNO.
; Parámetros que recibe:   La dirección de la cadena a leer en DS:DX.
;
; ********************************************************************************************
;
PedirFrases PROC NEAR
     PUSH DX                    ; Guardamos el contenido de DX.
     LEA DX, MsgPedirFrase      ; Ponemos en DS:DX, la dirección del texto a imprimir.
     CALL ImprimirCadena        ; Imprimimos la cadena.
     LEA DX, Frase              ; Ponemos en DS:DX la dirección de la frase a leer.
     CALL LeerCadena            ; Leemos la cadena.
     CALL Enter                 ; Producimos un ENTER.
     POP DX                     ; Recuperamos el contenido de DX.
     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 imprimir 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:     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:     Burbuja
```

```
; OBJETIVOS:            Realiza la ordenación de la cadena de caracteres por el método de la burbuja
; Parámetros que pasa:  NINGUNO.
; Parámetros que recibe: NINGUNO.
;; ********************************************************************************
;
Burbuja PROC NEAR
        PUSH CX                 ; Guardamos el contenido de los registros que vamos a emplear.
        PUSH AX
        LEA BX, Frase           ; Ponemos en DS:BX la dirección de la frase a ordenar.
        LEA SI, Frase           ; Ponemos en DS:SI la dirección de la frase a ordenar.
        INC BX                  ; Incrementamos BX en una posición.
        INC SI                  ; Incrementamos SI en dos posiciones.
        INC SI
        XOR CX, CX              ; Ponemos en CX el número de caracteres leídos
        MOV CL, [BX]            ; menos uno.
        PUSH CX                 ; Guardamos el contador.
        DEC CX
Bucle1:
        PUSH CX                 ; Guardamos los registros que emplearemos en cada vuelta del
        PUSH BX                 ; bucle.
        PUSH SI
Bucle2:
        INC BX                  ; BX apunta al primer carácter leído.
        INC SI                  ; SI apunta al segundo carácter leído.
        MOV AH, [BX]            ; Llevamos el contenido de BX a AH.
        MOV AL, [SI]            ; Llevamos el contenido de SI a AL.
        CMP AH, AL              ; Comparamos los dos caracteres.
        JB FinalBucle2          ; Si AH es menor que AL seguimos
        MOV [SI], AH            ; Intercambiamos AL y AH en la cadena
        MOV [BX], AL            ; de caracteres.
FinalBucle2:
        LOOP Bucle2             ; SI CX ≠ 0 seguimos con el bucle.
        POP SI                  ; Recuperamos los registros empleados en el bucle interno.
        POP BX
        POP CX
        LOOP Bucle1             ; SI CX ≠ 0 seguimos con el bucle.
        LEA BX, Frase           ; Ponemos en DS:BX la dirección de la cadena leída.
        INC BX                  ; Incrementamos en dos posiciones BX.
        INC BX
        POP CX                  ; Recuperamos el contador.
        ADD BX, CX              ; Posicionamos BX al final de la cadena.
        MOV CL, '$'             ; Ponemos un signo $ como final de la cadena.
        MOV [BX], CL
        POP AX                  ; Recuperamos el contenido que tenían los registros antes de
        POP CX                  ; llamar al procedimiento.
        RET
Burbuja ENDP
CODIGO ENDS
END Principal
```

5º) Realizar un programa en ensamblador que lea una cadena de 80 caracteres como máximo. Pida luego un carácter a buscar en la cadena y otro por el que se debe cambiar el carácter anterior si se encuentra.

Ejemplos

C:\> CAMBIA

Introduzca una frase de un máximo de 80 caracteres, por favor
La jaca Paca esta atada a la estaca
Introduzca el carácter a buscar, por favor
a
Por favor, introduzca el carácter por el que desea cambiar el anterior
o

La frase cambiada queda:

Lo joco Poco esto otodo o lo estoco

SOLUCIÓN:

```
; ********************************************************************************************
;
; A continuación definimos el segmento de datos.
;
; ********************************************************************************************
;
DATOS SEGMENT
    TerminarPrg         EQU   4C00h             ; Función de la INT 21h
    EscribeCadena       EQU   09h               ; Función de la INT 21h
    EscribeCaracter     EQU   02h               ; Función de la INT 21h
    LeeCadena           EQU   0Ah               ; Función de la INT 21h
    LeeCaracter         EQU   01h               ; Función de la INT 21h
    Espacio             EQU   20h               ; Código ASCII del espacio en blanco
    MsgPedirFrase       DB    'Introduzca una frase de un máximo de 80 caracteres, por favor', 10, 13, '$'
    MsgPedirCaracter    DB    'Introduzca el carácter a buscar, por favor', 10, 13, '$'
    MsgPedirCambio      DB    'Por favor, introduzca el carácter por el que desea cambiar el anterior', 10, 13, '$'
    MsgResultado        DB    'La frase cambiada queda: ', 10, 13, '$'
    MsgEnter            DB    10, 13, '$'       ; Cadena de texto que simula un ENTER
    Frase               DB    81, 0             ; Estructura para almacenar la frase.
                        DB    81 DUP (0)
    CaracterABuscar     DB    00h               ; Variable que contendrá el carácter a buscar.
    CaracterACambiar    DB    00h               ; Variable que contendrá el carácter a cambiar.
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.
;          PedirFrase:          Solicita  la  frase por teclado.
;          Enter:               Produce un salto de línea por pantalla.
;          ImprimirCadena:      Escribe la frase codificada por pantalla.
;          CambiarCaracter:     Cambia un carácter por otro.
; Parámetros que pasa:     NINGUNO.
; Parámetros que recibe:   NINGUNO.
;
; ********************************************************************************************
;
Principal PROC FAR
      MOV AX, DATOS
      MOV DS, AX
      CALL PedirFrase                ; Pedimos la frase
      CALL Enter                     ; Producimos un ENTER.
      LEA DX, MsgResultado           ; Ponemos en DS:DX el texto a imprimir.
      CALL ImprimirCadena            ; Imprimimos la cadena de caracteres.
      CALL Enter                     ; Producimos un ENTER.
      CALL CambiarCaracter           ; Cambiamos los caracteres.
      LEA DX, Frase                  ; Ponemos en DS:DX la dirección de la frase.
      ADD DX, 2h                     ; Incrementamos en dos la posición a la que apunta DX:
      CALL ImprimirCadena            ; Imprimimos la cadena.
      MOV AX, TerminarPrg            ; Terminamos el programa con la función 4C00h de
      INT 21h                        ; la INT 21h.
Principal ENDP
;
; ********************************************************************************************
;
; PROCEDIMIENTO:     PedirFrases.
; 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. También lee el
;                    carácter que se va a buscar y por el que se desea cambiar.
; Llama a los procedimientos:
;          ImprimirCadena:      Imprime una cadena de caracteres terminada en $.
;          LeerCadena:          Lee una cadena de caracteres.
;          Enter:               Produce un salto de línea por pantalla.
; Parámetros que pasa:          NINGUNO.
; Parámetros que recibe:        La dirección de la cadena a leer en DS:DX.
;; ********************************************************************************************
;
```

```
PedirFrase PROC NEAR
        PUSH DX                         ; Guardamos el contenido del registro DX.
        LEA DX, MsgPedirFrase           ; Ponemos en DS:DX la posición del texto a imprimir.
        CALL ImprimirCadena             ; Imprimimos la cadena.
        LEA DX, Frase                   ; Ponemos en DS:DX la posición de la cadena a leer.
        CALL LeerCadena                 ; Leemos la cadena
        CALL Enter                      ; Provocamos un ENTER.
        LEA DX, MsgPedirCaracter        ; Ponemos en DS:DX la posición del texto a imprimir.
        CALL ImprimirCadena             ; Imprimimos la cadena.
        MOV AH, LeeCaracter             ; Leemos un carácter por teclado mediante la función
        INT 21h                         ; 01h de la INT 21h
        MOV CaracterABuscar, AL         ; Guardamos el carácter leído en CaracterABuscar.
        CALL Enter                      ; Provocamos un ENTER.
        LEA DX, MsgPedirCambio          ; Ponemos en DS:DX la posición de la cadena a leer.
        CALL ImprimirCadena             ; Imprimimos la cadena.
        MOV AH, LeeCaracter             ; Leemos un carácter por teclado mediante la función
        INT 21h                         ; 01h de la INT 21h
        MOV CaracterACambiar, AL        ; Guardamos el carácter leído en CaracterACambiar.
        CALL Enter                      ; Provocamos un ENTER.
        POP DX                          ; Recuperamos el contenido de DX.
        RET
PedirFrase 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 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:     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:     CambiarCaracter.
; OBJETIVOS:         Busca el carácter pedido y lo cambia por el solicitado.
; Parámetros que pasa:     NINGUNO.
; Parámetros que recibe:   NINGUNO.
;
; ********************************************************************************************
;
CambiarCaracter PROC NEAR
      PUSH CX                  ; Guardamos el contenido de los registros que vamos a utilizar.
      PUSH AX
      LEA BX, Frase            ; Ponemos en DS:DX la dirección de la frase leída.
      INC BX                   ; Incrementamos BX en una posición.
      XOR CX, CX               ; Ponemos en CX el contador con el número de caracteres que
      MOV CL, [BX]             ; se han leído.
BucleCambiar:
      INC BX                   ; Incrementamos BX.
      MOV AL, [BX]             ; Ponemos en AL el carácter al que apunta BX.
      CMP AL, CaracterAbuscar  ; Comparamos con el carácter que buscamos.
      JNE SeguirBuscando       ; Si son diferentes sigo buscando.
      MOV AL, CaracterACambiar ; Ponemos en AL el carácter por el que deseamos cambiar el otro.
      MOV [BX], AL             ; Cambiamos el carácter
SeguirBuscando:
      LOOP BucleCambiar        ; SI CX ≠ 0 seguimos con el bucle.
      INC BX                   ; Incrementamos en uno la posición a la que apunta BX.
      MOV AL, '$'              ; Ponemos un signo de $ en esa posición de la cadena para que
      MOV [BX], AL             ; podamos imprimirla con la función 09h de la INT 21h.
      POP AX                   ; Recuperamos el contenido que tenían los registros antes de llamar
      POP CX                   ; al procedimiento.
      RET
CambiarCaracter ENDP
CODIGO ENDS
END Principal
```

6º) Realizar un programa en ensamblador que lea dos cadenas de un máximo de 80 dígitos hexadecimales e imprima como resultado la suma de los dos números.

Ejemplos:

C:\>SUMA

Introduzca el primer número hexadecimal, por favor (máximo 80 dígitos)
F0F0F10
Introduzca el segundo número hexadecimal, por favor (máximo 80 dígitos)
00000FF
La suma de los dos será:

F0F100F

SOLUCIÓN:

```
; ********************************************************************************************
;
; A continuación definimos el segmento de datos.
;
; ********************************************************************************************
;
DATOS SEGMENT
    TerminarPrg         EQU   4C00h             ; Función de la INT 21h.
    CorregirNumero      EQU   30h               ; Corrección de cifras entre 0 y 9.
    EscribeCadena       EQU   09h               ; Función de la INT 21h.
    EscribeCaracter     EQU   02h               ; Función de la INT 21h.
    LeeCadena           EQU   0Ah               ; Función de la INT 21h.
    DigitoMayor         EQU   09h               ; Dígito 9.
    CaracterMayor       EQU   39h               ; Código ASCII del 9.
    UNO                 EQU   31h               ; Código ASCII del 1.
    CERO                EQU   30h               ; Código ASCII del 0.
    Letra               EQU   07h               ; Corrección de cifras entre A y F.
    Numero              EQU   30h               ; Código ASCII del 0.
    MsgPedirNumero1     DB    'Introduzca el primer número hexadecimal, (máximo 80 dígitos)', 10, 13, '$'
    MsgPedirNumero2     DB    'Introduzca el segundo número hexadecimal, (máximo 80 dígitos)', 10, 13, '$'
    MsgResultado        DB    'La suma de los dos será: ', 10, 13, '$'
    MsgEnter            DB    10, 13, '$'       ; Cadena que simula un ENTER
    Numero1             DB    81, 0             ; Estructura para almacenar el primer número.
                        DB    81 DUP (0)
    Numero2             DB    81, 0             ; Estructura para almacenar el segundo número.
                        DB    81 DUP (0)
    Resultado           DB    83 DUP (0)        ; Estructura para almacenar el resultado.
                        DB    '$'
    ACARREOANTERIOR     DB    00h               ; Acarreo etapa anterior
    ACARREONUEVO        DB    00h               ; Acarreo etapa actual.
    MASCARAACARREO      DB    11110000B         ; Máscara para tomar el acarreo
    MASCARA4BITBAJOS    DB    0Fh               ; Máscara para quedarme con el nibble bajo.
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:
;           PedirFrases:              Solicita y lee por teclado dos números de un máximo de 80 cifras.
;           ConvertirLosNumeros:      Convierte de código ASCII a valor numérico.
;           ConvertirAFrase:          Convierte de valor numérico a código ASCII
;           Enter:                    Produce un salto de línea por pantalla.
;           Sumar:                    Realiza la suma de los dos números.
;           ImprimirResultado:        Escribe los dos números sumados por pantalla.
; Parámetros que pasa:      NINGUNO.
; Parámetros que recibe:    NINGUNO.
;
; ********************************************************************************************
;
Principal PROC FAR
      MOV AX, DATOS
      MOV DS, AX
      CALL PedirFrases                  ; Solicitamos los dos números.
      CALL Enter                        ; Producimos un salto de línea.
      CALL ConvertirLosNumeros          ; Convertimos de ASCII a número
      CALL Sumar                        ; Sumamos los dos números.
      CALL ConvertirAFrase              ; Convertimos de número a ASCII.
      CALL ImprimirResultado            ; Imprimimos el resultado.
      MOV AX, TerminarPrg               ; Terminamos el programa con la función 4C00h de la INT 21h
      INT 21h
Principal ENDP
```

```
;
; ********************************************************************************
;
; PROCEDIMIENTO:     PedirFrases.
; 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.
; Llama a los procedimientos:
;          ImprimirCadena:        Imprime una cadena de caracteres terminada en $.
;          LeerCadena:            Lee una cadena de caracteres.
;          Enter:                 Produce un salto de línea por pantalla.
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  La dirección de la cadena a leer en DS:DX.
;
; ********************************************************************************
;
PedirFrases PROC NEAR
     PUSH DX                        ; Guardamos el contenido del registro DX.
     LEA DX, MsgPedirNumero1        ; Ponemos en DS:DX la posición del texto a imprimir
     CALL ImprimirCadena            ; Imprimimos la frase.
     LEA DX, Numero1                ; Ponemos en DS:DX la posición en la que almacenar el
                                    ; primer número.
     CALL LeerCadena                ; Leemos el número
     CALL Enter                     ; Producimos un salto de línea.
     LEA DX, MsgPedirNumero2        ; Ponemos en DS:DX la posición del texto a imprimir.
     CALL ImprimirCadena            ; Imprimimos la frase.
     LEA DX, Numero2                ; Ponemos en DS:DX la posición en la que almacenar el
                                    ; segundo número.
     CALL LeerCadena                ; Leemos el números.
     POP DX                         ; Recuperamos el contenido del registro DX.
     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 imprimir 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:     ANumeroConvertir
; OBJETIVOS:         Convierte en valor numérico el código ASCII del número que pasamos en AL.
;                    Devuelve el también en AL el valor numérico ya convertido.
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  El código ASCII del carácter pasado en AL.
;
; ********************************************************************************************
;
ANumeroConvertir PROC NEAR
      CMP AL, CaracterMayor  ; Comparamos con el carácter '9'
      JBE EsNumero           ; Si es menor o igual salto a corregir el número.
      SUB AL, Letra          ; Restamos 7 al carácter pasado.
EsNumero:
      SUB AL, Numero         ; Restamos 30h al carácter pasado.
      RET
ANumeroConvertir ENDP
;
; ********************************************************************************************
;
; PROCEDIMIENTO:     ConvertirNumero.
; OBJETIVOS:         Convierte a valor numérico todas las cifras del número.
; Llama al procedimiento:
;          ANumeroCovertir:      Convierte en número el código ASCII que pasamos en AL..
; Parámetros que pasa:    Código ASCII del carácter en AL.
; Parámetros que recibe:  En DS:BX la dirección del número a transformar.
;
; ********************************************************************************************
;
```

```
ConvertirNumero PROC NEAR
        PUSH CX                 ; Guardamos el contenido de los registros que vamos a utilizar.
        PUSH AX
        INC BX                  ; Incrementamos en una posición BX.
        XOR CX, CX              ; Ponemos en CX el número de cifras leído.
        MOV CL, [BX]
BucleConvertir:
        INC BX                  ; Incrementamos BX en una posición.
        MOV AL, [BX]            ; Llevamos el contenido de BX al registro AL
        CALL ANumeroConvertir   ; Llamamos  para convertir el código ASCII a número.
        MOV [BX], AL            ; Guardamos el nuevo valor calculado.
        LOOP BucleConvertir     ; SI CX ≠ 0 seguimos con el bucle.
        POP AX                  ; Recuperamos el contenido que tenían los registros antes de llamar
        POP CX                  ; al procedimiento.
        RET
ConvertirNumero ENDP
;
; ********************************************************************************
;
; PROCEDIMIENTO:    AASCIIConvertir
; OBJETIVOS:        Convierte el valor numérico al código ASCII del número que pasamos en AL.
;                   Devuelve el también en AL el código ASCII ya convertido.
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  El valor numérico del carácter pasado en AL.
;
; ********************************************************************************
;
AASCIIConvertir PROC NEAR
        CMP AL, DigitoMayor     ; Comparamos con el dígito 9.
        JBE EsNum               ; Si es menor o igual es una cifra numérica, si no una cifra entre A y F.
        ADD AL, Letra           ; Sumamos 7 al valor numérico
EsNum:
        ADD AL, Numero          ; Sumamos 30h al valor numérico
        RET
AASCIIConvertir ENDP
;
; ********************************************************************************
;
; PROCEDIMIENTO:    ConvertirASCII.
; OBJETIVOS:        Convierte a código ASCII el valor numérico de todas las cifras del número.
; Llama al procedimiento:
;         AASCIIConvertir:      Convierte en código ASCII el número pasado en AL..
; Parámetros que pasa:    Valor numérico en AL.
; Parámetros que recibe:  En DS:SI la dirección del número a transformar.
;
; ********************************************************************************
;
ConvertirASCII PROC NEAR
        PUSH CX                 ; Guardamos el contenido que tenían de los registros.
        PUSH AX
        INC SI                  ; Incrementamos el registro SI en una posición
        XOR CX, CX
        MOV CL, [SI]            ; Ponemos en CX el número de caracteres leído más uno.
        INC CL
BucleConv:
        MOV AL, [BX]            ; Llevamos el contenido de BX a AL
        CALL AASCIIConvertir    ; Convertimos el número a código ASCII.
        MOV [BX], AL            ; Dejamos el resultado en la cadena.
        INC BX                  ; Incrementamos BX en una posición.
```

```
    LOOP BucleConv          ; SI CX ≠ 0 seguimos con el bucle.
    POP AX                  ; Recuperamos el contenido que tenían los registros antes
    POP CX                  ; de llamar al procedimiento.
    RET
ConvertirASCII ENDP
; ********************************************************************************
;
; PROCEDIMIENTO:    ConvertirLosNumeros
; OBJETIVOS:        Realiza la conversión de los dos números de código ASCII a valor numérico.
; Llama al procedimiento:
;       ConvertirNumero:    convierte de código ASCII a valor numérico una cadena de caracteres.
;
; Parámetros que pasa:    En DS:BX la posición de comienzo de la cadena a convertir.
; Parámetros que recibe:  NINGUNO.
;
; ********************************************************************************
;
ConvertirLosNumeros PROC NEAR
    PUSH BX                 ; Guardamos el contenido del registro BX.
    LEA BX, Numero1         ; Ponemos en DS:BX la posición de comienzo del primer número.
    CALL ConvertirNumero    ; Convertimos el primer número de código ASCII a número.
    LEA BX, Numero2         ; Ponemos en DS:BX la posición de comienzo del segundo número.
    CALL ConvertirNumero    ; Convertimos el segundo número de código ASCII a número.
    POP BX                  ; Recuperamos el contenido del registro BX.
    RET
ConvertirLosNumeros ENDP
; ********************************************************************************
;
; PROCEDIMIENTO:    ConvertirAFrase.
; OBJETIVOS:        Realiza la conversión de código ASCII a valor numérico.
; Llama al procedimiento:
;       ConvertirASCII:     convierte de valor numérico a código ASCII una cadena de caracteres.
;
; Parámetros que pasa:    En DS:BX la posición de comienzo de la cadena a convertir.
; Parámetros que recibe:  NINGUNO.
;
; ********************************************************************************
; ConvertirAFrase PROC NEAR
    PUSH BX                 ; Guardamos el contenido del registro BX.
    LEA BX, Resultado       ; Ponemos en DS:BX la posición de comienzo del resultado.
    LEA SI, Numero1         ; Ponemos en DS:SI la posición del primer número.
    CALL ConvertirASCII     ; Convertimos a ASCII el resultado.
    POP BX                  ; Recuperamos el contenido del registro BX.
    RET
ConvertirAFrase 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:     ImprimirResultado
; OBJETIVOS:         Imprime el resultado de la suma de los números..
; 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.
;
; ********************************************************************************************
;
ImprimirResultado PROC NEAR
      PUSH DX                  ; Guardamos el contenido de los registros que vamos a utilizar.
      PUSH AX
      PUSH SI
      LEA DX, MsgResultado     ; Ponemos en DS:DX la dirección del texto a imprimir.
      CALL ImprimirCadena      ; Imprimimos el texto.
      CALL Enter               ; Producimos un salto de línea.
      LEA DX, Resultado        ; Ponemos en DS:DX la dirección del resultado.
      LEA SI, Resultado        ; Ponemos en DS:SI la dirección del resultado.
      MOV AL, [SI]             ; Llevamos el contenido de SI a AL
      CMP AL, CERO             ; Comparamos con CERO.
      JNZ SigueImprimiendo     ; SI no es igual sigo imprimiendo.
      INC DX                   ; SI es cero incremento DX.
SigueImprimiendo:
      CALL ImprimirCadena      ; Imprimimos la cadena de caracteres.
      POP SI                   ; Recuperamos el contenido que tenían los registros antes de
      POP AX                   ; llamar al procedimiento.
      POP DX
      RET
ImprimirResultado ENDP
```

```
; ********************************************************************************
;
; PROCEDIMIENTO:     Sumar.
; OBJETIVOS:         Realiza la suma de los dos números.
; Parámetros que pasa:   NINGUNO.
; Parámetros que recibe: NINGUNO.
; ********************************************************************************
;
Sumar PROC NEAR
    PUSH SI                 ; Guardamos el contenido de los registros que vamos a emplear.
    PUSH BX
    PUSH CX
    PUSH AX
    PUSH DX
    PUSH DI
    XOR CX, CX              ; Ponemos el registro CX a cero.
    LEA BX, Numero1         ; Ponemos en DS:BX la dirección de comienzo del primer número.
    LEA SI, Numero2         ; Ponemos en DS:SI la dirección de comienzo del segundo número.
    INC BX                  ; Incrementamos BX en una posición.
    INC SI                  ; Incrementamos SI en una posición.
    XOR DX, DX              ; Ponemos DX z cero.
    MOV DL, [BX]            ; Llevamos el contenido de BX a DL
    ADD BX, DX              ; Sumamos BX más DX (longitud de la cadena). Nos ponemos al final.
    ADD SI, DX              ; Sumamos SI más DX (longitud de la cadena). Nos ponemos al final.
    MOV CL, DL              ; Ponemos en CL DL.
    LEA DI, Resultado       ; Ponemos en DI la dirección de comienzo del resultado.
    ADD DI, CX              ; Ponemos DI al final de la cadena resultado.
    PUSH DI                 ; Guardamos la dirección final del resultado
BucleSumar:
    MOV DL, [BX]            ; Llevamos a DL el contenido de BX.
    ADD DL, ACARREOANTERIOR     ; Sumamos DL con el posible acarreo de la suma anterior.
    TEST DL, MASCARAACARREO     ; Comprobamos si se ha producido un nuevo acarreo.
    JNZ SiHayAcarreo1           ; Si hay acarreo lo pongo.
    MOV ACARREONUEVO, 00h       ; Si no hay acarreo pongo AcarreoNuevo a cero.
    JMP SigueSumando            ; Sigo sumando las cifras.
SiHayAcarreo1:
    MOV ACARREONUEVO, 01h       ; SI hay acarreo pongo AcarreoNuevo a uno.
    AND DL, MASCARA4BITBAJOS    ; Me quedo con los cuatro bits inferiores del resultado.
SigueSumando:
    MOV DH, [SI]            ; Pongo en DH el contenido de la segunda cifra.
    ADD DH, DL              ; Sumo DH con DL
    TEST DH, MASCARAACARREO     ; Verifico si se produce acarreo.
    JNZ SiHayAcarreo2           ; Si hay acarreo actualizo el acarreo nuevo.
    JMP FinalBucleSumar         ; Si no hay acarreo salto al final del bucle.
SiHayAcarreo2:
    MOV ACARREONUEVO, 01h       ; SI hay acarreo pongo AcarreoNuevo a uno.
    AND DH, MASCARA4BITBAJOS    ; Me quedo con los cuatro bits inferiores del resultado.
FinalBucleSumar:
    MOV [DI], DH                ; Ponemos el resultado de las dos cifras sumadas.
    MOV DH, ACARREONUEVO        ; Ponemos en DH el AcarreoNuevo.
    MOV ACARREOANTERIOR, DH     ; Llevamos a AcarreoAnterior el nuevo acarreo.
    MOV ACARREONUEVO, 00h       ; Ponemos AcarreoNeuvo a cero para la siguiente iteración.
    DEC SI                      ; Decrementamos SI para acceder a la cifra anterior.
    DEC BX                      ; Decrementamos BX para acceder a la cifra anterior.
    DEC DI                      ; Decrementamos DI para acceder a la cifra anterior.
    LOOP BucleSumar             ; SI CX ≠ 0 seguimos con el bucle.
    MOV AL, ACARREOANTERIOR     ; Ponemos en AL AcarreoAnterior.
    MOV [DI], AL                ; Llevamos a la primera cifra del resultado ese acarreo.
    POP DI                      ; Recuperamos la posición del final del resultado
```

```
        INC DI                  ; y lo incrementamos en uno.
        MOV AL, '$'             ; ponemos en esa posición un $ para poder imprimir el
        MOV [DI], AL            ; resultado con la función 09h de la INT 21h.
        POP DI                  ; Recuperamos el contenido que tenían los registros antes
        POP DX                  ; de llamar al procedimiento.
        POP AX
        POP CX
        POP BX
        POP SI
        RET
Sumar ENDP
CODIGO ENDS
END Principal
```