SOLUCIONES COMENTADAS AL EXAMEN DE

LABORATORIO ESTRUCTURAS DE LOS COMPUTADORES

SEPTIEMBRE DE 1.999

(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. (**0,3 puntos**)
b) Si se le pasan menos de dos parámetros deberá sacar la ayuda (**0,3 puntos**)
c) Si se le pasan dos parámetros el primero será el nombre de usuario del sistema y el segundo la palabra clave de acceso al sistema. El nombre del archivo de claves y su ubicación se encuentra por seguridad en la variable de entorno ACCESO. Se pide:
   - Si el usuario y la clave son correctos deberá proporcionar un mensaje de bienvenida al sistema personalizado con el nombre del usuario y actualizar la variable Usuario con el nombre del usuario (**0,6 puntos**)
   - Si el usuario o la clave son erróneos se debería sacar un mensaje de error que indique que la clave o el usuario no son correctos y a continuación entrar en un bucle infinito (**0,3 puntos**)

Ejemplos:

C:\>LOGIN

El comando LOGIN permite la entrada al sistema WWW.ALCALA.ES
La sintaxis es:
"LOGIN [/? | *NombreUsuario PalabraClave]"*

C:\>
C:\>LOGIN Antonio hola
C:\> Bienvenido al sistema Antonio
C:\>LOGIN Pedro hola
C:\> Nombre de usuario o palabra clave no válidas.

NOTA 1: se supone que no existen nombres de usuarios iguales o parecidos (PEPE y PEPELUIS), y que un usuario no introducirá solamente una parte de la clave (HOLA y HO). Además el sistema deberá distinguir entre mayúsculas y minúsculas.

NOTA 2: el fichero de claves tiene la siguiente estructura (una por línea):

NombreUsuario1;PalabraClave1
NombreUsuario2;PalabraClave2
NombreUsuario3;PalabraClave3

SOLUCIÓN:

```
REM Desactivamos el ECHO de pantalla
@echo off

REM Si el usuario no pasa parámetros o pasa como parámetro /? sacamos
REM el mensaje de ayuda
IF "%1"=="" GOTO Ayuda
IF "%1"=="/?" GOTO Ayuda

REM Si no pasamos el segundo parámetro mostraremos la ayuda.
IF "%2"=="" GOTO Ayuda

REM Buscamos la cadena NombreUsuario;Clave en el fichero indicado en
REM la variable ACCESO
TYPE %ACCESO% | FIND "%1;%2" > NUL

REM Si no lo Encontramos no le permitiremos trabajar
IF ERRORLEVEL 1 GOTO NoAccede

REM Si el Usuario es correcto le damos la bienvenida y actualizamos la variable
REM Usuario con su nombre.
ECHO Bienvenido al sistema %1
SET Usuario=%1
GOTO Fin

REM Si el usuario no es válido entramos en un bucle infinito
```

```
:NoAccede
ECHO Nombre de usuario o palabra clave no válidas.
:Infinito
GOTO Infinito

REM Sacamos el mensaje de ayuda.
:Ayuda
ECHO Autorizaciones de acceso
ECHO Sintaxis:
ECHO "Login NombreUsuario Clave"
:Fin
```

2º) Indica qué es lo que hace el fichero BAT siguiente:

```
REM Desactiva el ECHO de pantalla
@echo off

REM Borra lo que hubiera en la variable dato
set dato=

REM Borra el archivo temporal fich si existe
if exist fich del fich

REM Entra en un bucle que lo hará 8 veces.
:bucle

REM Concatena al fichero fich un 8
echo 8 >> fich

REM Obliga al usuario a pulsar solamente las teclas 0 ó 1 sin mostrar
REM la pregunta por pantalla.
choice /C:01 >nul

REM Si pulso un 1 voy a EsUno
if errorlevel 2 goto EsUno

REM Si pulso un 0 concateno al final del dato un 0
set dato=%dato%0
goto FinBucle

REM Si pulso un 1 concateno al final del dato un 1
:EsUno
set dato=%dato%1
:FinBucle

REM Busco cuantas veces he introducido un 8 en el fichero fich
REM y busco que ese valor sea 8 en el segundo find.
type fich | find /c "8" | find "8"  >nul

REM Si no es 8, salto a bucle
if errorlevel 1 goto bucle

REM Si ya he introducido el byte lo muestro en pantalla
echo El dato es %dato%
```

SOLUCIÓN:

Permite introducir un byte en la variable dato.

3º) Dado el siguiente programa:

```
d segment
      n1 db 0FFh
      n2 dw 0000
d ends
p segment stack
      db 1024 dup (0)
p ends
c segment
assume cs:c, ds:d, ss:p
inicio:
      mov ax,d
      mov ds, ax

      xor bx, bx
      xor ax, ax
      mov al, n1
      mov bl, 0ah
      xor cx, cx
      Bucle:
      div bl
      cmp al, 0ah
      jb guardar
      push ax
      xor ah, ah
      INC DX
      jmp Bucle
      guardar:
        xor bx, bx
        add bl, al
        MOV DX, CX
        xor cx, cx
        mov cl, 4
        shl bx, cl
        add bl, ah
        cmp dx, 0
        je Final
        dec dx
        jz Ultimo
      Bucle2:
        shl bx, cl
        pop ax
        add bl, ah
        dec dx
        jnz Bucle2
      Ultimo:
        pop ax
        shl bx, cl
        add bl, ah
        Final:
        Mov n2, bx
      mov ah, 4ch
      int 21h
c ends
end inicio
```

Se pide:

a) Indicar que debería hace el programa si estuviese bien (**0,5 puntos**)
b) Indicar cuál es o cuáles son los errores del programa (**0,5 puntos**)
c) Corregir el programa para que funcione (**1 punto**)

SOLUCIÓN:

a) El programa lo que debería hacer es convertir el valor contenido en n1 (FFh) a su valor decimal 255 y dejar el resultado en n2.
b) Los errores se han marcado en letra itálica y mayúsculas en el código anterior. La instrucción INC DX lleva la cuenta del número de divisiones por diez que se han realizado y se necesita para saber cuantos valores hemos introducido en la pila.
   La línea MOV DX, CX sobra ya que cambiaría el valor de DX y no sabríamos cuantos números hemos introducido en la pila.
c) La corrección del programa consiste en incluir la instrucción INC DX dentro del bucle Bucle y eliminar la línea MOV DX, CX que hace que perdamos el número de dígitos guardados en la pila.

4º) Se desea realizar un programa en ensamblador que lea una cadena de un máximo de 10 dígitos entre 0 y 9 (se supone que se introducen correctamente y que el cero es par) y presente el siguiente menú:

1.- Escribir los pares
2.- Escribir los impares
3.- Salir

Para ello se pide:

a) Un procedimiento que lea la cadena de 10 dígitos así como la definición de los datos que se crean necesarios para su funcionamiento (**0,4 puntos**)
b) Un procedimiento que lea el menú y permita escoger entre una de las opciones, así como la definición de los datos que se crean necesarios para que funcione (**0,6 puntos**)
c) Un procedimiento que escriba los números pares (**0,5 puntos**)
d) Un procedimiento que escriba los números impares (**0,5 puntos**)

Ejemplos:

C:\> PARIMPAR

Introduce un máximo de 10 dígitos entre 1 y 9.
1122334455

1.- Escribir los pares
2.- Escribir los impares
3.- Salir

1

2244

SOLUCIÓN:

Una posible implementación es la siguiente. La hemos dividido por apartados y luego colocamos el código junto al final.

a) En el segmento de datos estará definida:

```
MaximoMasUno    DB    11        ;Estructura para leer los 10 dígitos
NumDigitos      DB    0
Digitos         DB    11 DUP (0)
```

Y en el segmento de código:

```
PedirDigitos PROC NEAR
    PUSH DX                 ; Guardamos el contenido de los registros.
    PUSH AX

    LEA DX, MSgPEdir        ; Indicamos en DS:DX la dirección de la cadena a
escribir.
    CALL ImprimirCadena     ; Imprimimos la cadena

    LEA DX, MaximoMasUno    ; Indicamos en DS:DX la dirección de la cadena a leer.
    CALL LeeCadena          ; Leemos la cadena.


    POP AX                  ; Recuperamos el contenido de los registros.
    POP DX
    RET
PedirDigitos ENDP

LeeCadena PROC NEAR
    PUSH AX                 ; Guardamos el registro que vamos a utilizar.
    MOV AH, LeerCadena      ; Imprimimos la cadena mediante la función 09h
    INT 21h                 ; de la INT 21h.
    POP AX                  ; Recuperamos el contenido que tenía el registro.
    RET
LeeCadena ENDP

ImprimirCadena PROC NEAR
    PUSH AX                 ; Guardamos el registro que vamos a utilizar.
    MOV AH, EscribirCadena  ; Imprimimos la cadena mediante la función 09h
    INT 21h                 ; de la INT 21h.
    POP AX                  ; Recuperamos el contenido que tenía el registro.
    RET
ImprimirCadena ENDP
```

b) En el segmento de datos se definirá:

```
MsgMenu     DB    10,13, '1.- Escribir los pares.',10,13
            DB    '2.- Escribir los impares.', 10, 13
            DB    '3.- Salir.',10, 13, '$'
```

Y los procedimientos serán:

```
ImprimirCadena PROC NEAR
      PUSH AX                 ; Guardamos el registro que vamos a utilizar.
      MOV AH, EscribirCadena  ; Imprimimos la cadena mediante la función 09h
      INT 21h                 ; de la INT 21h.
      POP AX                  ; Recuperamos el contenido que tenía el registro.
      RET
ImprimirCadena ENDP

Menu PROC NEAR
   PUSH AX            ; Guardamos el contenido de los registros
   PUSH DX

Bucle:
   LEA DX, MsgMenu          ; Ponemos en DS:DX la dirección de MsgMenu.
   CALL ImprimirCadena      ; Imprimimos el menú.

   MOV AH, LeerCaracter     ; Función de la INT 21h que deja el código ASCII del
   INT 21h                  ; carácter leído en AL

   CMP AL, '1'              ; Si es uno escribimos los pares.
   JE OpPares
   CMP AL, '2'              ; Si es dos escribimos los impares.
   JE OpImpares
   CMP AL, '3'              ; Si es tres salimos del programa.
   JE OpSalir

   JMP Bucle
OpPares:
   CALL ImprimirPares       ; Imprimir pares.
   JMP FinMenu
OpImPares:
   CALL ImprimirImpares     ; Imprimir impares.
   JMP FinMenu
OpSalir:
FinMenu:
   POP DX             ; Recuperamos el contenido de los registros.
   POP AX
   RET
Menu ENDP
```

c) El procedimiento para escribir los números pares es:

```
ImprimirPares PROC NEAR
    PUSH SI                 ; Guardamos el contenido de los registros.
    PUSH CX
    PUSH AX
    PUSH DX
    XOR SI, SI              ; Inicializamos SI y CX a cero.
    XOR CX, CX
    MOV CL, NumDigitos      ; Ponemos en CX cuantos dígitos hemos leído
    LEA DX, MsgPares        ; Sacamos el mensaje de Pares.
    CALL ImprimirCadena
    BuclePares:
      TEST Digitos[SI], 1   ; Si el bit menos significativo está a uno es para
      JNZ FinBuclePares     ; Si no es cero es un impar.
      MOV AH, EscribirCaracter  ; Función de la INT 21h.
      MOV DL, Digitos[SI]   ; Ponemos en DL el código ASCII del carácter a
imprimir.
      INT 21h
      FinBuclePares:
          INC SI            ; Pasamos al siguiente dígito leído
          LOOP BuclePares
   POP DX                   ; Recuperamos el contenido de los registros
   POP AX
   POP CX
   POP SI
   RET
ImprimirPares ENDP
```

d) El procedimiento para imprimir los números impares es:

```
ImprimirImpares PROC NEAR
    PUSH SI                 ; Guardamos el contenido de los registros.
    PUSH CX
    PUSH AX
    PUSH DX
    XOR SI, SI              ; Inicializamos a cero SI y CX
    XOR CX, CX
    MOV CL, NumDigitos      ; Ponemos en CX el número de dígitos leídos
    LEA DX, MsgImpares      ; Imprimimos el mensaje de impares
    CALL ImprimirCadena
    BucleImpares:
      TEST Digitos[SI], 1   ; Si el bit menos significativo está a 1 es impar.
      JZ FinBucleImpares    ; Si es cero es par.
      MOV AH, EscribirCaracter ; Función de la INT 21h.
      MOV DL, Digitos[SI]   ; Ponemos en DL el código ASCII del dígito a imprimir
      INT 21h
      FinBucleImpares:
          INC SI            ; Pasamos al siguiente dígito
          LOOP BucleImpares
   POP DX                   ; Recuperamos el contenido de los registros.
   POP AX
   POP CX
   POP SI
   RET
ImprimirImpares ENDP
```

El código comentado junto con otras definiciones queda:

```
; ********************************************************************************
;
; A continuación definimos el segmento de datos.
;
; ********************************************************************************
;
DATOS SEGMENT
      TerminarPRG     EQU     4Ch         ; Función de la INT 21h
      LeerCaracter    EQU     1           ; Función de la INT 21h
      LeerCadena      EQU     0Ah         ; Función de la INT 21h
      EscribirCaracter EQU    2           ; Función de la INT 21h
      EscribirCadena  EQU     9h          ; Función de la INT 21h

      MsgPedir        DB      'Introduzca diez dígitos máximo entre 0 y 9, por favor',10,13,'$'
      MaximoMasUno    DB      11          ;Estructura para leer los 10 dígitos
      NumDigitos      DB      0
      Digitos         DB      11 DUP (0)

      MsgMenu         DB      10,13, '1.- Escribir los pares.',10,13
                      DB      '2.- Escribir los impares.', 10, 13
                      DB      '3.- Salir.',10, 13, '$'
      MsgPares        DB      10,13, 'Los números pares son: $'
      MsgImPares      DB      10,13, 'Los números impares son: $'
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
;
;
; ********************************************************************************
;
; A continuación definimos el procedimiento PRINCIPAL que llama a los siguientes procedimientos:
;           PedirDigitos:     Solicita y lee por teclado un máximo de diez dígitos entre 0 y 9.
;           Menu:             Presenta el menú solicitado.
; Parámetros que pasa:      NINGUNO.
; Parámetros que recibe:    NINGUNO.
;
; ********************************************************************************
;
```

```
PRINCIPAL PROC FAR
	ASSUME CS:CODIGO, DS:DATOS, SS:PILA
	MOV AX, DATOS
	MOV DS, AX

	CALL PedirDigitos		; Pedimos la cadena de dígitos.
	CALL Menu			; Mostramos y realizamos el menú

	MOV AH, TerminarPrg	; Terminamos el programa con la función 4C00h de la INT 21h
	INT 21h

PRINCIPAL ENDP
;
; ********************************************************************************************
;
; PROCEDIMIENTO:	PedirDígitos.
; OBJETIVOS:		Lee un máximo de 10 dígitos entre 0 y 9.  Deja la cadena leída en
;			la estructura Digitos. Emplea la función 0Ah de la INT 21h.
; Llama a los procedimientos:
;		ImprimirCadena:		Imprime una cadena de caracteres terminada en $.
;		LeeCadena:		Lee una cadena de caracteres.
; Parámetros que pasa:	NINGUNO.
; Parámetros que recibe:	La dirección de la cadena a leer y a escribir en DS:DX.
;
; ********************************************************************************************
;
PedirDigitos PROC NEAR
	PUSH DX			; Guardamos el contenido de los registros que vamos a usar.
	PUSH AX

	LEA DX, MSgPEdir		; Indicamos en DS:DX la dirección de la cadena a escribir.
	CALL ImprimirCadena	; Imprimimos la cadena

	LEA DX, MaximoMasUno	; Indicamos en DS:DX la dirección de la cadena a leer.
	CALL LeeCadena		; Leemos la cadena.


	POP AX			; Recuperamos el contenido de los registros.
	POP DX
	RET
PedirDigitos ENDP
;
; ********************************************************************************************
;
; PROCEDIMIENTO:	LeeCadena.
; OBJETIVOS:		Lee una cadena de caracteres función 0Ah de la INT 21h.
; Parámetros que pasa:	NINGUNO.
; Parámetros que recibe:	La  dirección de la cadena a imprimir en DS:DX.
;
; ********************************************************************************************
;
LeeCadena PROC NEAR
	PUSH AX			; Guardamos el registro que vamos a utilizar.
	MOV AH, LeerCadena	; Imprimimos la cadena mediante la función 09h
	INT 21h			; de la INT 21h.
	POP AX			; Recuperamos el contenido que tenía el registro.
	RET
LeeCadena 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, EscribirCadena  ; Imprimimos la cadena mediante la función 09h
      INT 21h                 ; de la INT 21h.
      POP AX                  ; Recuperamos el contenido que tenía el registro.
      RET
ImprimirCadena ENDP
;
; ********************************************************************************
;
; PROCEDIMIENTO:    Menu
; OBJETIVOS:        Presenta y controla el menú solicitado. Llama al procedimiento:
;
;         ImprimirCadena:       Imprime una cadena de caracteres terminada en $.
;
; Parámetros que pasa:      En DS:DX la dirección de la cadena a imprimir.
; Parámetros que recibe:    NINGUNO.
;
; ********************************************************************************
;
Menu PROC NEAR
    PUSH AX             ; Guardamos el contenido de los registros
    PUSH DX

Bucle:
    LEA DX, MsgMenu         ; Ponemos en DS:DX la dirección de MsgMenu.
    CALL ImprimirCadena     ; Imprimimos el menú.

    MOV AH, LeerCaracter    ; Función de la INT 21h que deja el código ASCII del
    INT 21h                 ; carácter leído en AL

    CMP AL, '1'             ; Si es uno escribimos los pares.
    JE OpPares
    CMP AL, '2'             ; Si es dos escribimos los impares.
    JE OpImpares
    CMP AL, '3'             ; Si es tres salimos del programa.
    JE OpSalir

    JMP Bucle
OpPares:
    CALL ImprimirPares      ; Imprimir pares.
    JMP FinMenu
OpImPares:
    CALL ImprimirImpares    ; Imprimir impares.
    JMP FinMenu
OpSalir:
FinMenu:
    POP DX              ; Recuperamos el contenido de los registros.
    POP AX
    RET
Menu ENDP
```

```
;
; ********************************************************************************
;
; PROCEDIMIENTO:    ImprimirPares.
; OBJETIVOS:        Imprime los números pares de la cadena mediante la
;                   función 02h de la INT 21h. Llama al procedimiento:
;
;         ImprimirCadena:     Imprime una cadena de caracteres terminada en $.
;
; Parámetros que pasa:      En DS:DX la dirección de la cadena a imprimir.
; Parámetros que recibe:    NINGUNO.
;
; ********************************************************************************
;
ImprimirPares PROC NEAR
    PUSH SI                 ; Guardamos el contenido de los registros.
    PUSH CX
    PUSH AX
    PUSH DX
    XOR SI, SI              ; Inicializamos SI y CX a cero.
    XOR CX, CX
    MOV CL, NumDigitos      ; Ponemos en CX cuantos dígitos hemos leído
    LEA DX, MsgPares        ; Sacamos el mensaje de Pares.
    CALL ImprimirCadena
    BuclePares:
      TEST Digitos[SI], 1   ; Si el bit menos significativo está a uno es para
      JNZ FinBuclePares     ; Si no es cero es un impar.
      MOV AH, EscribirCaracter  ; Función de la INT 21h.
      MOV DL, Digitos[SI]   ; Ponemos en DL el código ASCII del carácter a imprimir.
      INT 21h
      FinBuclePares:
          INC SI            ; Pasamos al siguiente dígito leído
          LOOP BuclePares
   POP DX                   ; Recuperamos el contenido de los registros
   POP AX
   POP CX
   POP SI
   RET
ImprimirPares ENDP
;
; ********************************************************************************
;
; PROCEDIMIENTO:    ImprimirImpares.
; OBJETIVOS:        Imprime los números impares de la cadena mediante la
;                   función 02h de la INT 21h. Llama al procedimiento:
;
;         ImprimirCadena:     Imprime una cadena de caracteres terminada en $.
;
; Parámetros que pasa:      En DS:DX la dirección de la cadena a imprimir.
; Parámetros que recibe:    NINGUNO.
;
; ********************************************************************************
;
ImprimirImpares PROC NEAR
    PUSH SI                 ; Guardamos el contenido de los registros.
    PUSH CX
    PUSH AX
    PUSH DX
    XOR SI, SI              ; Inicializamos a cero SI y CX
    XOR CX, CX
    MOV CL, NumDigitos      ; Ponemos en CX el número de dígitos leídos
```

```
    LEA DX, MsgImpares          ; Imprimimos el mensaje de impares
    CALL ImprimirCadena
    BucleImpares:
      TEST Digitos[SI], 1       ; Si el bit menos significativo está a 1 es impar.
      JZ FinBucleImpares        ; Si es cero es par.
      MOV AH, EscribirCaracter  ; Función de la INT 21h.
      MOV DL, Digitos[SI]       ; Ponemos en DL el código ASCII del dígito a imprimir
      INT 21h
      FinBucleImpares:
            INC SI              ; Pasamos al siguiente dígito
            LOOP BucleImpares
   POP DX                       ; Recuperamos el contenido de los registros.
   POP AX
   POP CX
   POP SI
   RET
ImprimirImpares ENDP
CODIGO ENDS
END PRINCIPAL
```