SOLUCIONES AL EXAMEN DE

LABORATORIO DE ESTRUCTURAS DE LOS COMPUTADORES

CURSO 95/96. SEPTIEMBRE 1.996

Parte de Ensamblador.

1º) Leer un número por teclado y mostrar en pantalla el complemento a 1 y el complemento a 2 del número leído. Ejemplo: Si se introduce el número 17 el programa deberá mostrar: E8 (complemento a 1) y E9 (complemento a 2).

Solución:

```
;
; A continuación declaramos el segmento de datos.
;
Datos SEGMENT
  LeerC                    EQU 01h
  Escribir                 EQU 02h
  He_Leido_Numero          EQU 3Ah
  Es_Numero                EQU 0Ah
  Corrige_Numero           EQU 30h
  Corrige_Letra            EQU 07h
  Salto_De_Linea           EQU 0Ah
  EnterC                   EQU 0Dh
  Nibble_Bajo              EQU 0Fh
  Numero_Leido              DB  00h
  Complemento_A_1           DB  00h
  Complemento_A_2           DB  00h
Datos ENDS
;
Pila SEGMENT STACK
      DB 1024 DUP (?)
Pila ENDS
;
; Una posible implementación del segmento de código es la siguiente:
;
Codigo SEGMENT
  ASSUME CS:Codigo, DS:Datos, SS:Datos   ;Inicialización de  los
                                         ;segmentos del programa.
Inicio PROC FAR
  MOV AX, Datos
  MOV DS, AX
  XOR CX, CX

  CALL Leer
  CALL CorregirANum
  MOV Numero_Leido, AL
  CALL Leer
  CALL CorregirANum
  MOV DL, Numero_Leido
  MOV CL, 04h
  SHL DL, CL
  ADD DL, AL
  MOV Numero_Leido, DL
  CALL Enter
```

```
   CALL Operar
   MOV DL, Complemento_A_1
   CALL Imprimir
   CALL Enter
   MOV DL, Complemento_A_2
   CALL Imprimir
   CALL Terminar
   RET
Inicio ENDP
;
Leer PROC NEAR
;
; Leemos un número que se supone comprendido entre 01h y 0Fh por teclado.
;
   XOR AX, AX
   MOV AH, LeerC
   INT 21h
   RET
Leer ENDP
;
CorregirANum PROC NEAR
;
; Convertimos el  código  ASCII  en el número  de veces que se debe mostrar
; en pantalla. Para  ello, debemos de  comprobar  si el número se corresponde
; con  el  código ASCII  de  una  letra (y  le   restaremos 37h para  obtener  su
; valor numérico)o un número (en cuyo caso se  le restara 30h). El  valor  con
; el  que   debemos   comparar  es  3Ah, ya  que   el  código  ASCII  que   se
; corresponde  con los números va desde el 30h hasta el 39h. Y almacenamos
; el valor corregido en la variable Numero_Leido.
;
   CMP AL, He_Leido_Numero
   JLE Era_Numero
   SUB AL, Corrige_Letra
Era_Numero:
   SUB AL, Corrige_Numero
   RET
CorregirAnum ENDP
;
CorregirALetra PROC NEAR
;
; Convertimos el  código  ASCII  en el número  de veces que se debe mostrar
; en pantalla. Para  ello, debemos de  comprobar  si el número se corresponde
; con una  letra (y  le   sumaremos 37h para  obtener  su código ASCII.
; o un número (en cuyo caso se  le sumaremos 30h). El  valor  con
; el  que   debemos   comparar  es  0Ah.
;
   CMP DL, Es_Numero
   JLE Era_Numero2
   ADD DL, Corrige_Letra
```

```
Era_Numero2:
   ADD DL, Corrige_Numero
   RET
CorregirALetra ENDP
;
Operar PROC NEAR
;
; Ponemos ahora en AL el verdadero  valor del  número que  se ha leído por
; teclado. Tenemos que  escribir el complemento a 1 y a 2 del número.
; Deberemos de indicar que deseamos escribir y no leer.
;
   XOR AX, AX
   XOR CX, CX
   MOV AL, Numero_Leido
   NOT AL
   MOV Complemento_A_1, AL
   NOT AL
   NEG AL
   MOV Complemento_A_2, AL
   RET
Operar ENDP
;
Enter PROC NEAR
; Ahora saltamos de línea para que el resultado lo escriba en la línea
; siguiente. Para ello deberemos de escribir un 0Ah y después un 0Dh.
;
   XOR AX, AX
   XOR DX, DX
   MOV AH, Escribir
   MOV DL, Salto_De_Linea
   INT 21h
   MOV DL, EnterC
   INT 21h
   RET
Enter ENDP
;
Imprimir PROC NEAR
    MOV AH, Escribir
   XOR CX,CX
   MOV BL, DL
   MOV CL, 04h
   SHR DL, CL
   CALL CorregirALetra
   INT 21h
   AND BL, 0Fh
   MOV DL, BL
   Call CorregirAletra
   INT 21h
   RET
```

```
Imprimir ENDP
;
Terminar PROC NEAR
;
; Finalmente deberemos de indicar al sistema operativo que se ha terminado
; el programa. Para ello se emplea la interrupción 21h y se almacena en AH
; el valor 4Ch.
;
  XOR AX, AX
  MOV AH, 4Ch
  INT 21h
  RET
Terminar ENDP
Codigo ENDS
END inicio
```

2º) Leer por pantalla 15 números de cuatro cifras hexadecimales por teclado y mostrar en pantalla el mayor y el menor número que se ha leído. Se supone que existe un procedimiento que se encarga de leer un número de cuatro cifras hexadecimales por teclado llamado LeeCuatro y otro que escribe un número de cuatro cifras hexadecimales en pantalla llamado EscribeCuatro. Ambos emplean el registro AX, de tal forma que: LeeCuatro devuelve el número leído por teclado en AX; y EscribeCuatro imprime en pantalla el número que se encuentre en AX. La forma de llamarlos en el código ensamblador es: **CALL LeeCuatro** y **CALL EscribeCuatro**. Ejemplo si se leen los números 1111, 2222, 3333, 4444, 5555, 6666, 7777, 8888, 9999, AAAA, BBBB, CCCC, DDDD, EEEE, y FFFF, el programa deberá mostrar los números FFFF y 1111.

Solución:

```
;
; A continuación declaramos el segmento de datos.
;
Datos SEGMENT
  Leer                    EQU 01h
  Escribir                EQU 02h
  He_Leido_Numero         EQU 3Ah
  Es_Numero               EQU 0Ah
  Corrige_Numero          EQU 30h
  Corrige_Letra           EQU 07h
  Salto_De_Linea          EQU 0Ah
  EnterC                  EQU 0Dh
  Mayor_Numero_Leido      DW  0000h
  Menor_Numero_Leido      DW  0FFFFh
  Cadena                  DB  5, 0, 0, 0, 0, 0, 0, '$'
  LeerCadena              EQU 0Ah
  EscribeCadena           EQU 09h
Datos ENDS
;
Pila SEGMENT STACK
```

```
        DB 1024 DUP (?)
Pila ENDS
;
; Una posible implementación del segmento de código es la siguiente:
;
Codigo SEGMENT
  ASSUME CS:Codigo, DS:Datos, SS:Pila   ;Inicialización de   los
                                        ;segmentos del programa.
Inicio PROC FAR
  MOV AX, Datos
  MOV DS, AX
;
;  Leemos primero dos números y los ordenamos.
;
  XOR AX, AX
  XOR CX, CX
  CALL LeeCuatro
  MOV Mayor_Numero_Leido, AX
  MOV Menor_Numero_Leido, AX
  XOR AX, AX
  CALL LeeCuatro
  CMP AX, Mayor_Numero_Leido
  JB Era_Menor              ; Para no tener en cuenta el signo
  MOV Mayor_Numero_Leido, AX
  JMP Sigue
Era_Menor:
  CMP AX, Menor_Numero_Leido
  JA Sigue                  ; Para no tener en cuenta el signo
  MOV Menor_Numero_Leido, AX
Sigue:
;
;  Como tenemos que leer 15 números por teclado, emplearemos un bucle que irá desde
;  el numero 12 decimal (0Ch) hasta el número 0 (00h), ya que dos ya los hemos leído.
;  También emplearemos el procedimiento LeeCuatro que devuelve el número leído, tal
;  y como nos indican en el enunciado.
;  En la ejecución del bucle debemos comprobar si es mayor o menor que el número que
;  hemos leído antes.
;
  XOR AX, AX
  XOR CX, CX
  MOV CL, 0Ch
Bucle_De_Lectura:
  CALL LeeCuatro
  CMP AX, Mayor_Numero_Leido
  JB Ver_Si_Es_El_Minimo      ; Para no tener en cuenta el signo
  MOV Mayor_Numero_Leido, AX
  JMP Siguiente
Ver_Si_Es_El_Minimo:
  CMP AX, Menor_Numero_Leido
```

```
   JA Siguiente              ; Para no tener en cuenta el signo
   MOV Menor_Numero_Leido, AX
Siguiente:
   LOOP Bucle_De_Lectura
;
; Escribimos el resultado con el procedimiento EscribeCuatro.
;
  Call Enter
  XOR AX, AX
  MOV AX, Menor_Numero_Leido
  CALL EscribeCuatro
  CALL Enter
  XOR AX, AX
  MOV AX, Mayor_Numero_Leido
  CALL EscribeCuatro
;
; Finalmente deberemos de indicar al sistema operativo que se ha terminado
; el programa. Para ello se emplea la interrupción 21h y se almacena en AH
; el valor 4Ch.
;
   XOR AX, AX
   MOV AH, 4Ch
   INT 21h
;
; Con todo lo anterior ya hemos terminado el segmento de código.
;
   RET
Inicio ENDP
;
;
Enter PROC NEAR
        PUSH AX
        PUSH DX
        MOV AH, Escribir
        MOV DL, Salto_De_Linea
        INT 21h
        MOV DL, EnterC
        INT 21h
        POP DX
        POP AX
        RET
Enter ENDP
;
```

```
CorregirANum PROC NEAR
;
; Convertimos el  código  ASCII  en el número  de veces que se debe mostrar
; en pantalla. Para  ello, debemos de  comprobar  si el número se corresponde
; con  el  código ASCII  de  una  letra (y  le   restaremos 37h para  obtener  su
; valor numérico)o un número (en cuyo caso se  le restara 30h). El  valor  con
; el  que   debemos   comparar  es  3Ah, ya  que   el  código  ASCII  que   se
; corresponde  con los números va desde el 30h hasta el 39h. Y almacenamos
; el valor corregido en la variable Numero_Leido.
;
  CMP AL, He_Leido_Numero
  JLE Era_Numero
  SUB AL, Corrige_Letra
Era_Numero:
  SUB AL, Corrige_Numero
  RET
CorregirAnum ENDP
;
CorregirALetra PROC NEAR
;
; Convertimos el  código  ASCII  en el número  de veces que se debe mostrar
; en pantalla. Para  ello, debemos de  comprobar  si el número se corresponde
; con una  letra (y  le   sumaremos 37h para  obtener  su código ASCII.
; o un número (en cuyo caso se  le sumaremos 30h). El  valor  con
; el  que   debemos   comparar  es  0Ah.
;
  CMP AL, Es_Numero
  JL Era_Numero2
  ADD AL, Corrige_Letra
Era_Numero2:
  ADD AL, Corrige_Numero
  RET
CorregirALetra ENDP
;
LeeCuatro PROC NEAR
        PUSH BX
        PUSH CX
        PUSH DX
        MOV AH, LeerCadena
        LEA DX, Cadena
        INT 21h
        LEA BX, Cadena
        INC BX
        XOR CX, CX
        MOV CL, [BX]
        Bucle2:
          INC BX
          MOV AL, [BX]
          Call CorregirANum
```

```
            MOV [BX], AL
            LOOP Bucle2
        XOR CX, CX
        MOV CL, 04h
        LEA BX, Cadena
        INC BX
        INC BX
        MOV DH, [BX]
        SHL DH, CL
        INC BX
        ADD DH, [BX]
        INC BX
        MOV DL, [BX]
        XOR CX, CX
        MOV CL, 04h
        SHL DL, CL
        INC BX
        ADD DL, [BX]
        MOV AX, DX
        POP DX
        POP CX
        POP BX
        RET
LeeCuatro ENDP
;
EscribeCuatro PROC NEAR
        PUSH BX
        PUSH CX
        PUSH DX
        LEA BX, Cadena
        INC BX
        INC BX
        XOR DX, DX
        XOR CX, CX
        MOV DH, AH
        MOV CL, 04
        SHR DH, CL
        MOV [BX], DH
        INC BX
        MOV DL, AH
        AND DL, 0Fh
        MOV [BX], DL
        INC BX
        XOR DX, DX
        XOR CX, CX
        MOV DH, AL
        MOV CL, 04
        SHR DH, CL
        MOV [BX], DH
```

```
        INC BX
        MOV DL, AL
        AND DL, 0Fh
        MOV [BX], DL
        LEA BX, Cadena
        INC BX
        XOR CX, CX
        MOV CL, [BX]
        Bucle:
          INC BX
          MOV AL, [BX]
          Call CorregirALetra
          MOV [BX], AL
          LOOP Bucle
        MOV AH, EscribeCadena
        LEA DX, Cadena
        INC DX
        INC DX
        INT 21h
        POP DX
        POP CX
        POP BX
        RET
EscribeCuatro ENDP
Codigo ENDS
END Inicio
```

3º) Suponiendo que en la pila tenemos una expresión aritmética que otro programa se encargará de evaluar, se pide realizar un programa en ensamblador que indique si los paréntesis están bien colocados (devolverá un 1) o no (devolverá un 0). Es decir, si existe el mismo número de paréntesis abiertos que cerrados o si no se empieza por un paréntesis cerrado en vez de por uno abierto. Además, el primer elemento de la pila indica el número de elementos que se deberán leer de la pila (sin contarle a él). Por simplicidad, cada elemento de la pila contiene un único valor, operador aritmético o paréntesis. Ejemplos:

((2 + 3) x 5) devolverá un 1.

(((2 + 3) x 5) devolverá un 0 (hay un paréntesis abierto de más).

(2 + 3)) devolverá un 0 (hay un paréntesis cerrado de más).

) 2 + 3 (devolverá un 0 (empieza por un paréntesis cerrado).

( ) 2 + 3 NO SE CONTEMPLARÁN CASOS COMO ÉSTE.

Solución:

```
;
;
; A continuación declaramos el segmento de datos.
;
Datos SEGMENT
  Escribir                  EQU 02h
  Expresion_Correcta        EQU 31h
  Expresion_Erronea         EQU 30h
  Es_Parentesis_Abierto     EQU 28h
  Es_Parentesis_Cerrado     EQU 29h
  Salto_De_Linea            EQU 0Ah
  Enter                     EQU 0Dh
Datos ENDS
;
Pila SEGMENT STACK
        DB 1024 DUP (?)
Pila ENDS
;
; Una posible implementación del segmento de código es la siguiente:
;
Codigo SEGMENT
  ASSUME CS:Codigo, DS:Datos, SS:Pila  ; Inicialización de   los
                                       ; segmentos del programa.
Inicio PROC FAR
  MOV AX, Datos
  MOV DS, AX
  CALL Meter
;
; Leemos el primer elemento de la pila que indicará cuantos elementos debemos leer en
; la pila. La idea es que con ese valor hagamos un bucle.
;
  XOR CX, CX
  XOR DX, DX
  POP CX
;
;  Haremos un bucle en el que sacaremos los elementos de la pila y verificaremos que
; los paréntesis abiertos y cerrados estén bien. Una forma posible es:
;      - Cada vez que encontremos un paréntesis abierto sumamos uno a un registro.
;      - Cada vez que encontremos un paréntesis cerrado restaremos uno a un registro.
;      - Si en algún momento existe un valor negativo, quiere decir que hasta ese punto
; han aparecido más paréntesis cerrados que abiertos. La expresión estará mal.
Bucle_De_Lectura_De_La_Pila:
  POP AX
  CMP AL, Es_Parentesis_Abierto
  JNE Mira_Si_Es_Cerrado
  INC DX
  JMP Seguir_Bucle
Mira_Si_Es_Cerrado:
```

```
   CMP AL, Es_Parentesis_Cerrado
   JNE Seguir_Bucle
   DEC DX
Seguir_Bucle:
   CMP DX, 00h
   JL Escribe_Respuesta_Erronea
   LOOP Bucle_De_Lectura_De_La_Pila
;
;  Al salir del bucle, deberemos comprobar lo siguiente.
;       - Si el valor de DX es positivo indica que la expresión está mal. Hay más paréntesis
;         abiertos que cerrados.
;       - Si el valor de DX es 0 indica que la expresión es correcta. Hay el mismo numero
;           de paréntesis abiertos que de cerrados.
;
   CMP DX, 00h
   JE Escribe_Respuesta_Correcta
Escribe_Respuesta_Erronea:
   MOV DL, Expresion_Erronea
   JMP Escribe_Respuesta
Escribe_Respuesta_Correcta:
   MOV DL, Expresion_Correcta
   XOR AX, AX
Escribe_Respuesta:
   XOR DH, DH
   MOV AH, Escribir
   INT 21h
;
; Finalmente deberemos de indicar al sistema operativo que se ha terminado
; el programa. Para ello se emplea la interrupción 21h y se almacena en AH
; el valor 4Ch.
;
   XOR AX, AX
   MOV AH, 4Ch
   INT 21h
   RET
Inicio ENDP
Codigo ENDS
END Inicio
```

Parte de MS-DOS.

1º)Escribir un fichero BAT, llamado **ejecuta** al que se le da el nombre de un fichero ejecutable, mire si se encuentra o no en ese directorio, y en caso de no encontrarlo lo ensambla, lo linka y lo ejecuta. Si lo encuentra, debe ejecutar el programa.
Ejemplo: Sea contenido del directorio C:\TMP

```
 El volumen de la unidad C es MS-DOS_6
 El número de serie del volumen es 20BC-A8C0
 Directorio de C:\TMP
.          <DIR>        07-20-96  10:44a
..         <DIR>        07-20-96  10:44a
1                  0 07-20-96  10:47a
TODOASCI EXE       24,408 01-09-94   9:51p
ESCRIBE  ASM          570 07-26-88  12:00a
       6 archivo(s)        52,357 bytes
                   525,795,328 bytes libres
```

ejecuta todoasci ejecutará el programa todasci, ya que se encuentra en el directorio.
ejecuta escribe ensamblará, linkará y ejecutará el programa escribe.asm

Solución:

```
@ECHO OFF
  IF NOT EXIST %1.EXE GOTO NoExiste
  %1
  GOTO FIN
:NoExiste
  MASM %1
  LINK %1
  %1
:Fin
```

2º)Trabajamos para una empresa que se dedica a llevar la contabilidad para otras cuatro empresas, y nos piden automatizar el proceso de copia de seguridad creando un disco por cada empresa. El programa debe actuar de la siguiente manera:

Copia [emp1] [emp2] ...

Si introducimos más de cuatro empresas, hará la copia de seguridad de las cuatro primeras y despreciará el resto, mostrando el mensaje de error correspondiente. El mensaje de ayuda lo mostrará cuando no le introduzcamos parámetros.

Para hacer la copia de seguridad deberá llamar a un archivo de extensión .com con igual nombre que la empresa, el archivo que se encuentra en el directorio: c:\copias.

Deberá mostrar un mensaje de error si una de las cuatro primeras empresas no existe; saliéndose posteriormente del programa.

Solución:

```
@ECHO OFF
:BucleCopia
  IF NOT(%5)==() GOTO MuchasEmpresas
  C:\COPIAS\%1
  SHIFT
  IF "%1"=="" GOTO Fin
  GOTO BucleCopia
:MuchasEmpresas
  ECHO Se han introducido más de cuatro empresas.
:Fin
```

3º)Normalmente cuando se instala una nueva aplicación para Windows, se copian una serie de ficheros nuevos en los directorios C:\WINDOWS y C:\WINDOWS\SYSTEM y además se modifican los ficheros WIN.INI y SYSTEM.INI con la nueva información.

Se pide escribir un fichero BAT, llamado **diferent**, que tras haber realizado la instalación de una nueva aplicación para Windows realice las siguientes operaciones:

a) Cree un directorio a partir del C:\ETC\INF con el nombre del programa que se instaló.
b) Compare los ficheros WIN.INI y SYSTEM.INI nuevos con los anteriores que se encuentran en el directorio C:\ETC\WININI y dejen las diferencias en un fichero llamado **WININI.DIF**.
c) Comparar el contenido de los directorios C:\WINDOWS y C:\WINDOWS\SYSTEM con el contenido anterior (fichero FICHEROS.TXT) situado en el directorio C:\ETC\WININI y dejará la información en un fichero llamado **FICHEROS.DIF**.
   NOTA: Tanto el fichero **WININI.DIF** como **FICHEROS.DIF** deberán guardarse en un directorio con el nombre del programa que se instaló.
   Ejemplo: **diferent WINWORD** realizará las operaciones siguientes:
   a) Creará el directorio **C:\ETC\INF\WINWORD**.
   b) Creará el fichero **WININI.DIF** dentro del directorio **C:\ETC\INF\WINWORD** con las diferencias de los ficheros WIN.INI y SYSTEM.INI.
   c) Creará el fichero **FICHEROS.DIF** dentro del directorio **C:\ETC\INF\WINWORD** con las diferencias en el contenido de l directorio C:\WINDOWS y C:\WINDOS\SYSTEM.

Solución:

```
@ECHO OFF
MD C:\ETC\INF\%1
FC C:\ETC\WININI\WIN.INI C:\WINDOWS\WIN.INI >>
C:\ETC\INF\%1\WININI.DIF
FC C:\ETC\WININI\SYSTEM.INI C:\WINDOWS\SYSTEM.INI >>
C:\ETC\INF\%1\WININI.DIF
DIR C:\WINDOWS /S >> TEMPORAL
FC TEMPORAL C:\ETC\WININI\FICHEROS.TXT >>
C:\ETC\INF\%1\FICHEROS.DIF
DEL TEMPORAL
```