



Problema 1 parcial. (Febrero mañana).

(7 Puntos).

Dado el siguiente programa:

```
INCLUDE macros.mac
MODEL SMALL

MACRO Mac1

STACK 200H
DATASEG
var1 DB ?
var2 DB ?
var3 DB 13 DUP (?)
var4 DW ?
var5 DB 1024 DUP (?)
var6 DB 20H,2CH,2EH,3BH,3AH,9,10,13,0
var7 DB 0
var8 DD 0
var9 DD 0
Msg1 ...
Msg2 ...
Msg3 ...
Msg4 ...
Err1 ...
Err2 ...
cr_lf DB 13,10,0
Tabla DB '0123456789'

eti1: READ_HANDLE var4,var5,1024 ;B
      JC Error2 ;B
      OR AX,AX ;B
      JZ eti6 ;B
      MOV CX,AX ;B
      LEA SI,var5 ;B
eti2: LEA DI,var6 ;B
      MOV AL,[SI] ;B
eti3: CMP AL,[DI] ;B
      JE eti4 ;B
      INC DI ;B
      CMP BYTE PTR [DI],0 ;B
      JNE eti3 ;B
      INC var9 ;B
      CMP var7,0 ;B
      JNE eti5 ;B
      INC var8 ;B
      MOV var7,1 ;B
      JMP eti5 ;B
eti4: MOV var7,0 ;B
eti5: INC SI ;B
      LOOP eti2 ;B
      JMP eti1 ;B
eti6:

FRAGMENTO 1
      CLOSE_HANDLE var4
      Mac1 var8,Msg3

FRAGMENTO 2
      END_PROCESS 0

Error1:
Error2:
END

CODESEG
P386N
EXTRN UDTA:NEAR
STARTUPCODE
DISP_STRINGZ Msg1 ;A
GET_STRING 13,var1 ;A
DISP_STRINGZ cr_lf ;A
XOR BX,BX ;A
MOV BL,var2 ;A
MOV var3[BX],0 ;A
OPEN_HANDLE var3,0 ;A
JC Error1 ;A
MOV var4,AX ;A
```



Problema 1 parcial. (Febrero mañana).

(Cont).

Se pide:

- A. Codificar la macro *Mac1*, que tiene dos parámetros, para que realice lo siguiente:
- Mostrar por pantalla un mensaje cuyo nombre de variable es el segundo parámetro.
 - Convertir el primer parámetro, que es un entero sin signo de 32 bits, en una cadena ASCII en base 10 y con un tamaño mínimo de un carácter, y dejarla en *var3*.
 - Visualizar por pantalla la cadena numérica resultante y colocar el cursor al principio de la siguiente línea.
- (1 pto.)
- B. Codificar *Fragmentos 1y 2* para que realicen lo siguiente:
- Fragmento 1* debe calcular el tamaño del fichero que está abierto, y llamar a la macro *Mac1* para que muestre por pantalla el mensaje *Msg2* y el tamaño del fichero.
 - Fragmento 2* debe realizar lo siguiente:
 - Comprobar si *var8* es distinto de cero; en ese caso, dividir *var9* entre *var8* y dejar el cociente en *var9*, y si no (si *var8* es cero), dejar *var9* como estaba.
 - A continuación, independientemente de que se haya tenido que realizar o no la división, llamar a la macro *Mac1* para que muestre por pantalla el mensaje *Msg4* y el valor numérico *var9*.
- (1 pto.)
- C. ¿Qué hace el programa en aquellas instrucciones que contienen como comentario la letra *A*? Se debe indicar de forma clara qué hace el programa, y no describir que hace cada instrucción de forma aislada. También se indicará qué información se almacena en las variables *var1*, *var2*, *var3*, y *var4*. (1 pto.)
- D. ¿Qué hace el programa en aquellas instrucciones que contienen como comentario la letra *B*? Se debe responder con las mismas condiciones expuestas en el apartado anterior. Debe indicarse qué información se almacena en las variables *var5*, *var6*, *var7*, *var8*, y *var9*. Téngase en cuenta que los caracteres *20H*, *2CH*, *2EH*, *3AH*, *3BH*, *9*, *10* y *13* se corresponden con el espacio, la coma, el punto, los dos puntos, el punto y coma, el tabulador, el avance de línea y el retorno de carro respectivamente. (1,75 ptos.)
- E. ¿Qué hace el programa en conjunto?. Debe ponerse valores significativos a *Msg1*, *Msg2*, *Msg3* y *Msg4*. Indica que mensajes aparecerían en pantalla durante la ejecución del programa si el usuario tecleara al pedirle la cadena que se le pide el valor: *a.txt*, y en ese fichero se tuvieran almacenadas las dos primeras líneas de este programa (“**INCLUDE macros.mac**”,10,13, “**MODEL SMALL**”,10,13). (1, 75 ptos.)
- F. Escribe el código que debe aparecer junto a cada etiqueta de tratamiento de errores (*Error1*: y *Error2*:). Se debe mostrar en cada caso el mensaje de error que corresponda (*Err1* y *Err2*), cerrar los ficheros que estén abiertos cuando se produzca el error, y poner un código de retorno del programa equivalente al número de error producido. (Si se sale por *Error1*, se debe poner código de retorno 1, y así para los demás casos). Codificar también los mensajes de error *Err1* y *Err2*. (0,5 ptos.)

NOTA: Cada apartado debe resolverse en las zonas indicadas. Esta primera hoja y el reverso de las siguientes son para anotaciones en sucio.



Problema 1 parcial. (Febrero mañana).

(Cont).

APARTADO A)

```
Mac1    MACRO    num,msg
        DISP_STRINGZ msg
        PUSH    num
        PUSH    WORD PTR 10
        PUSH    WORD PTR 1
        PUSH    OFFSET Tabla
        PUSH    OFFSET var3
        CALL    UDTA
        DISP_STRINGZ var3
        DISP_STRINGZ cr_1f
        ENDM
```

APARTADO B)

```
; Fragmento 1
eti6:   MOVE_PTR  var4,0,0,2    ;C
        SHL     EDX,16         ;C
        MOV     DX,AX          ;C
        Mac1    EDX,Msg2       ;C

; Fragmento 2
        CMP     var8,0         ;C
        JZ     eti7           ;C
        MOV     EAX,var9       ;C
        XOR     EDX,EDX        ;C
        DIV     var8           ;C
        MOV     var9,EAX       ;C
eti7:   Mac1    var9,Msg4      ;C
```



Problema 1 parcial. (Febrero mañana).

(Cont).

APARTADO C)

Se muestra Msg1 pidiéndole al usuario que teclee el nombre de un fichero (13 caracteres máximo incluyendo el retorno de carro). La cadena tecleada se convierte a ASCIIZ y se intenta abrir el fichero. Si se produce un error se salta a Error1 y si se abre con éxito, se guarda el handle del fichero en la variable Var4.

Var1, Var2 y Var3 forman la estructura requerida por la función GET_STRING (Var1 contendrá un 13 —número máximo de caracteres—; Var2 el número de caracteres de la cadena tecleada sin contar el retorno de carro, y Var3 la cadena en sí).

APARTADO D)

Se intenta leer todo el fichero en trozos de 1 KB (1.024 bytes). La información leída se almacena en el buffer var5. Si existe algún problema se salta a Error2. Posteriormente se comprueba que se haya leído al menos un byte, porque si no se ha leído ninguno, es que hemos llegado al final del fichero y nos salimos por eti6.

Se almacena el número de bytes leídos en CX, que servirá como contador de las iteraciones de eti2. Es decir se recorrerá el bucle una vez por cada carácter leído. También se hace que SI apunte al buffer de lectura del fichero.

Una vez en el bucle se apunta con DI a la variable var6 que contiene un conjunto de caracteres que se pueden considerar como separadores de palabras en un texto.

Se lee un carácter del buffer del fichero y se comprueba en el bucle eti3 si el carácter leído es alguno de los separadores de var6.

Si es un separador, var7 (que funciona como un switch) se pone a 0, Se incrementa SI (puntero al buffer del fichero) y se vuelve al bucle eti2 para repetir el mismo proceso con el siguiente carácter del fichero.

Si el carácter leído no es un separador, se incrementa var9 (que es un contador de caracteres que no aparecen en la lista de separadores). Si el switch vale 1, significará que el carácter anterior no es un separador, con lo que no se hace otra cosa que repetir el bucle después de incrementar SI para tratar el siguiente carácter del fichero. Si el switch valía 0, significa que el carácter anterior era un separador, y se incrementa var8 que funciona como un contador de palabras. Como var7 (el switch) se inicializa a 0 al declararlo, esta cuenta de palabras también vale para la primera palabra del fichero. Finalmente el switch se pone a 1 para destacar que ya no estamos al comienzo de una palabra al leer el carácter siguiente.



Problema 1 parcial. (Febrero mañana).

(Cont).

APARTADO E)

El programa solicita al usuario que teclee un nombre de fichero, y se muestra el tamaño del fichero, el número de palabras que contiene y el número medio de letras que hay en cada palabra.

```
Msg1    DB      "Introduce nombre de fichero: ",0
Msg2    DB      "Número total de caracteres en el fichero: ",0
Msg3    DB      "Número de palabras contenidas en el fichero: ",0
Msg4    DB      "Número medio de caracteres por palabra: ",0
```

Ejemplo de ejecución:

```
Introduce nombre de fichero: a.txt
Número total de caracteres en el fichero: 33
Número de palabras contenidas en el fichero: 5
Número medio de caracteres por palabra: 5
```

APARTADO F)

```
Err1    DB      'Error abriendo fichero', 13, 10, 0
Err2    DB      'Error de lectura en fichero', 13, 10, 0
```

```
Error1: DISP_STRINGZ Err1
        END_PROCESS 1
Error2: CLOSE_HANDLE var4
        DISP_STRINGZ Err2
        END_PROCESS 2
```



Problema 2 parcial. (Febrero mañana).

(3 Puntos).

En las siguientes preguntas de tipo test, sólo existe una respuesta correcta. Cada pregunta tiene un valor de 0,3 pts. Cada respuesta incorrecta descuenta 0,1 pts. La pregunta que esté en blanco no sumará ni restará puntos.

Notas: • Se debe poner una X sobre la letra de la tabla "SOLUCIÓN" (que encontrará en la última hoja) correspondiente a la respuesta correcta.

• La tabla "CALIFICACIÓN" no deberá utilizarla. Es para su posterior calificación.

• No se tendrán en cuenta las respuestas que no se encuentren en dicha tabla "SOLUCIÓN".

1) La instrucción MOV [EDX+8*EBP-6],AX

- a) Es incorrecta, porque el segundo operando debiera ser un registro de 32 bits.
- b) Es incorrecta, porque el registro EBP no puede usarse como registro índice.
- c) Es correcta y se usa SS para el cálculo de la dirección física.
- d) Es correcta y se usa DS para el cálculo de la dirección física.

2) En un 8086, durante el proceso de atención a una interrupción en modo real se lee el valor 21H en la dirección física 000CFH

- a) Se trata de la interrupción 34H y la rutina de servicio está en la dirección 21XXH:XXXXH.
- b) Se trata de la interrupción 21H y la rutina de servicio está en la dirección XX21H:XXXXH.
- c) Se trata de la interrupción 33H y la rutina de servicio está en la dirección XXXXH:XX21H.
- d) Se trata de la interrupción 33H y la rutina de servicio está en la dirección 21XXH:XXXXH.

3) La ejecución de la llamada a la macro MOVE_PTR Hand1, -1, -7, 1 ...

- a) ... deja el puntero al principio del fichero.
- b) ... avanza el puntero 64 K posiciones aproximadamente.
- c) ... retrocede el puntero 7 posiciones.
- d) ... produciría un error, pues no es una manera correcta de llamar a esa macro.



Problema 2 parcial. (Febrero mañana).

(Cont).

4) Los contenidos de la posición de memoria 034EH y siguientes son : 02H, 0F4H, 34H, 12H, 00H, 0F0H, 0F1H. Sabiendo que esta zona de memoria corresponde con la tabla de vectores de interrupción, que comienza en la dirección 00000, indicar la dirección completa de una rutina de atención a una interrupción:

- a) 1234H:F402H.
- b) F000H:1234H.
- c) F1F0H:0012H.
- d) 1200H:F0F1H.

5) Si el offset de la variable long3 es 4000H y BX contiene 13H, la instrucción:

```
MOV long3[BX],7:
```

- a) Carga el valor 7 en la dirección DS:4013H
- b) No es correcta
- c) Carga el valor 13H en la dirección DS:4000H
- d) Carga el valor de 20H en la dirección DS:4000H

6) Si los contenidos de los siguientes registros son:

```
SS = 7000H, DS = A000H, BP = 7CA4H y SI = 2B92H
```

El operando [BP+SI+9AH] hace referencia a un dato que se encuentra en la dirección:

- a) AA7D0H
- b) 7A8D0H
- c) 7A7D0H
- d) AA8D0H

7) Sea el siguiente fragmento de programa:

```
NUM DB '128',0
.....
LEA AX,NUM
PUSH AX
PUSH WORD PTR 10
CALL ATUD
```

Sabiendo que devuelve en AX un valor positivo, éste debería ser:

- a) 0
- b) 1
- c) 2
- d) 4



Problema 1 parcial/extraordinaria. (Febrero tarde).

(7/4 Puntos).

Dado el siguiente programa:

INCLUDE macros.mac

Declaración de la MACRO Macro1
Declaración de la MACRO Macro2

```

MODEL          SMALL
STACK          200H
DATASEG
msg1           DB          .....
msg2           DB
msgerr1        DB          .....
msgerr2        DB          .....
msgerr3        DB          .....
var1           DB          ?
var2           DB          ?
var3           DB          80 DUP(?)
var4           DW          ?
var5           DD          ?
var6           DB          40 DUP(?)
var7           DB          5 DUP(?)
var8           DB          ?
var9           DB          "0123456789"
var10          DB          7 DUP(?)
var11          DB          10,13,0
CODESEG
P386N
EXTRN          .....
               STARTUPCODE          ;C
Macro1        msg1,80                ;C
OPEN_HANDLE   var3,0                 ;C
JC            error1                 ;C
MOV           var4,AX                ;C
Macro1        msg2,40                ;C
etiql1:      READ_HANDLE   var4,var5,50 ;C
JC            error2                 ;C
OR            AX,AX                  ;C
JZ            etiql4                 ;C
CMP           AX,50                  ;C
JNE           error3                 ;C
XOR           BX,BX                  ;D
bucle1:      MOV           AL,var3[BX] ;D
OR            AL,AL                  ;D
JZ            etiql2                 ;D
CMP           AL,var6[BX]            ;D
JNE           etiql1                 ;D
INC           BX                     ;D
JMP           bucle1                 ;D
etiql2:      Macro2        var5,6     ;E
DISP_STRINGZ var10             ;E
DISPLAY_CHAR " "               ;E
XOR           BX,BX                 ;E
bucle2:      MOV           AL,var6[BX] ;E
OR            AL,AL                 ;E
JZ            bucle3                 ;E
DISPLAY_CHAR AL                 ;E
INC           BX                     ;E
JMP           bucle2                 ;E
bucle3:      DISPLAY_CHAR " "       ;E
INC           BX                     ;E
CMP           BX,40                 ;E
JNE           bucle3                 ;E
XOR           EBX,EBX               ;E
MOV           BL,var8               ;F
Macro2        EBX,3                 ;F
MOV           var10[4],0            ;F
MOV           AL,var10[2]           ;F
MOV           var10[3],AL           ;F
MOV           var10[2],","          ;F
CMP           var10[0],"0"          ;F
JNE           etiql3                 ;F
MOV           var10[0]," "          ;F
etiql3:      DISP_STRINGZ  var10     ;F
DISPLAY_CHAR " "                   ;F
MOV           var8,0                 ;F
DISP_STRINGZ var7                   ;F
DISP_STRINGZ var11                  ;F
JMP           etiql1                 ;F
etiql4:      CLOSE_HANDLE  var4     ;C
END_PROCESS  0                       ;C
error1:      DISP_STRINGZ  msgerr1   ;C
END_PROCESS  1                       ;C
error2:      CLOSE_HANDLE  var4     ;C
DISP_STRINGZ msgerr2                 ;C
END_PROCESS  2                       ;C
error3:      CLOSE_HANDLE  var4     ;C
DISP_STRINGZ msgerr3                 ;C
END_PROCESS  3                       ;C
END

```



Problema 1 parcial/extraordinaria. (Febrero tarde).

(Cont).

Sabiendo que el programa anterior maneja un fichero de alumnos que incluye de cada uno de ellos el número de expediente, el nombre completo, el grupo al que pertenece y la nota obtenida en la asignatura de microprocesadores (en este mismo orden), se pide:

- A) Codificar la MACRO Macro1, que tiene 2 pseudoparámetros, para que realice las siguientes acciones:
- Visualizar el mensaje cuyo nombre de variable es el primer parámetro de la macro
 - Leer de teclado una cadena de caracteres, cuyo tamaño máximo (contando el retorno de carro final) es el segundo parámetro de la macro, y dejarla en la variable **var3**.
 - Convertir la cadena recibida a ASCIIZ.
- (0,8/0,4 ptos.)
- B) Codificar la MACRO Macro2, que tiene también 2 pseudoparámetros, para que convierta el entero sin signo de 32 bits, que recibe como primer parámetro, en una cadena ASCIIZ expresada en decimal. La cadena resultante tiene que quedar en la variable **var10** y debe tener como mínimo el número de caracteres indicado en el segundo pseudoparámetro. Esta macro puede llamar a una rutina de la librería **conv32**, pero entonces debe completarse también la sentencia **EXTRN** que está al principio del segmento de código. (0,9/0,5 ptos.)
- C) ¿Qué hace el fragmento identificado con el comentario “;C”? ¿Qué variable o variables escribe la sentencia **READ_HANDLE var4,var5,50**? ¿Qué causa le hace saltar a **eti4**? Completar los mensajes **msg1**, **msgerr1**, **msgerr2** y **msgerr3** de forma explícita (no valen expresiones genéricas tales como, por ejemplo, “Introducir cadena”). (1,6/0,9 ptos.)
- Nótese que en el programa hay dos trozos separados que tienen el comentario “;C”: el primero al principio del código y el otro al final. Deben incluirse ambos en la explicación.*
- D) ¿Qué hace el fragmento identificado con el comentario “;D”? ¿Cuándo se sale del bucle por **eti1**, y cuando por **eti2**? ¿Qué es la segunda cadena que se introduce por teclado en el fragmento anterior (en **Macro1 msg2,40**)?. Completar **msg2**, también de forma explícita. (0,6/0,4 ptos.)
- E) ¿Qué hace el fragmento identificado con el comentario “;E”? ¿Qué visualiza por pantalla?. ¿Qué hacen, en concreto, los bucles **bucle2** y **bucle3**? (1,0/0,6 ptos.)
- F) ¿Qué hace el fragmento identificado con el comentario “;F”? ¿Qué manipulación realiza con **var10**? indicar en qué formato está especificada en el fichero, y en qué formato se visualiza. ¿Por qué o para qué coloca un 0 en **var8**? ¿Qué más visualiza por pantalla este fragmento?. (1,2/0,7 ptos.)
- G) ¿Qué hace el programa en su conjunto? ¿Cuántos registros del listado visualiza?. (0,9/0,5 ptos.)

NOTAS:

- Para la resolución del problema pueden ser de interés los siguientes códigos ASCII:
Salto de línea y retorno de carro 10 y 13 respectivamente
NULL 0
- Cada apartado debe resolverse en las zonas indicadas de las páginas siguientes. No se valorará lo que se escriba sobre el programa de la hoja anterior.

**Problema 1 parcial/extraordinaria. (Febrero tarde).****(Cont).****SOLUCIÓN****Apartado A:*****Declaración de Macro1***

```
Macro1 MACRO mensaje, limite
DISP_STRINGZ mensaje
GET_STRING limite, var1
MOV BL, var2
XOR BH, BH
MOV var3[BX], 0
DISP_STRINGZ var1
ENDM
```

Apartado B:***Declaración de Macro2***

```
Macro2 MACRO valor, minimo
PUSH valor
PUSH WORD PTR 10
PUSH WORD PTR minimo
PUSH OFFSET var9
PUSH OFFSET var10
CALL udta
ENDM
```

Sentencia EXTRN:

```
EXTRN udta:NEAR
```

Apartado C:***Explicación del fragmento***

El fragmento ejecuta el código inicial (STARTUPCODE), pide por teclado el pathname del fichero de alumnos y lo intenta abrir; si hay error, salta a error1, visualiza msgerr1 y termina con código de retorno 1; caso contrario, guarda el handle en var4 y pide una segunda cadena de 40 caracteres como máximo, cuyo significado y uso se verá en el siguiente fragmento. A continuación lee 50 bytes (todo un registro) del fichero, y los deja en las variables var5 a var8 ambas inclusive, que, según el enunciado serán presumiblemente el número de expediente, el nombre completo, el grupo y la nota, respectivamente. Si se produce error, se salta a error2, se visualiza msgerr2, se cierra el fichero y se sale con código de error 2, y si no, se comprueba si el número de caracteres leídos es 0 o es un número mayor que 0 pero menor que 50; en el primer caso, es que se ha llegado al final del fichero, y en ese caso se salta a etiq4, se cierra el fichero y se sale sin error, y en el segundo, se salta a error3, se visualiza msgerr3, se cierra el fichero y se sale con código de retorno 3. Si se han leído 50 bytes, se continúa.



Problema 1 parcial/extraordinaria. (Febrero tarde).

(Cont).

Mensajes

```
msg1      DB "Introduzca el path del fichero: ",0
msgerr1   DB "Error al abrir fichero",10,13,0
msgerr2   DB "Error al leer fichero",10,13,0
msgerr3   DB "Error: registro truncado",10,13,0
```

Apartado D:

Explicación del fragmento

Compara carácter a carácter el segundo campo del registro leído (var6), que presumiblemente es el nombre completo del alumno, con la segunda cadena que se ha pedido por teclado (en Macro1 msg2,40), hasta llegar al NULL de esta última; si se encuentra alguna desigualdad, se vuelve a etiql para leer un nuevo registro, y en cambio, si se llega al NULL de la cadena introducida por teclado, es decir, si el nombre del alumno comienza por esa cadena, se sale del bucle bucle1 saltando a etiql2.

Mensajes

```
msg2      DB "Teclee el inicio del nombre del alumno: ",0
```

Apartado E:

Explicación del fragmento

Este fragmento comienza convirtiendo el primer campo del registro leído (var5), que es una doble word binaria (presumiblemente el número de expediente) a una cadena ASCII decimal con 6 caracteres como mínimo, y la visualiza por pantalla seguida de un espacio. A continuación, el bucle2 visualiza el segundo campo del registro carácter a carácter hasta llegar al NULL final (el nombre completo, que debe ser una cadena ASCII de menos de 40 caracteres), llevando la cuenta en BX del número de caracteres escritos. Seguidamente el bucle3 visualiza espacios por pantalla incrementando también BX hasta que valga 40. De esta forma, el número de caracteres escritos entre bucle2 y bucle3 es siempre 40, y por lo tanto, el siguiente campo que se visualice quedará siempre encolumnado.



Problema 1 parcial/extraordinaria. (Febrero tarde).

(Cont).

Apartado F:

Explicación del fragmento

Empieza convirtiendo `var8`, que es el último campo del registro leído (la nota, que es un byte en binario), en una cadena ASCII decimal de 3 caracteres, para lo cual comienza convirtiendo el byte (BL) a una doble word (EBX). A continuación, se realizan dos manipulaciones a la cadena resultante (`var10`): se inserta una coma decimal entre el carácter de la derecha y el anterior, y además, si la cadena comienza por 0, se convierte éste en un espacio. Así, por ejemplo, un 7,3 de nota se guarda en el fichero como 01001001, que es la representación binaria del 73 decimal, y cuando se convierte a ASCII queda "073",0, y después de la manipulación queda " 7,3",0. Seguidamente, se visualiza esta cadena, un espacio, el tercer campo (`var7`, que es el grupo al que pertenece el alumno) y un salto de línea, y se salta a `etiql` para leer un nuevo registro. La única explicación que tiene la instrucción que coloca un 0 en `var8` es poner el NULL terminador de `var7`, que presumiblemente no debe estar incluido en el fichero.

Apartado G:

Explicación del fragmento

El programa pide por teclado el nombre del fichero de alumnos y el inicio del nombre de un alumno, y se visualiza por pantalla un listado encolumnado de todos los alumnos cuyo nombre comience por esas letras, incluyendo en él el número de expediente, el nombre completo, la nota que ha sacado y el grupo al que pertenece.

Número de registros que se visualizan

Depende de la cadena que se introduzca como inicio del nombre, el programa puede no encontrar ninguno, o sólo uno (por ejemplo, si hemos introducido como cadena el nombre completo de un alumno concreto), o bien varios (si introducimos "Pérez", el listado incluirá a todos los alumnos cuyo primer apellido sea Pérez), o incluso todos, si introducimos una cadena nula (sólo pulsamos ENTER).



Problema 2 parcial. (Febrero tarde).

(3 Puntos).

En las siguientes preguntas de tipo test, sólo existe una respuesta correcta. Cada pregunta tiene un valor de 0,3 pts. Cada respuesta incorrecta descuenta 0,1 pts. La pregunta que esté en blanco no sumará ni restará puntos.

Notas: • Se debe poner una X sobre la letra de la tabla "SOLUCIÓN" (que encontrará en la última hoja) correspondiente a la respuesta correcta.

• La tabla "CALIFICACIÓN" no deberá utilizarla. Es para su posterior calificación.

• No se tendrán en cuenta las respuestas que no se encuentren en dicha tabla "SOLUCIÓN".

1) La instrucción MOV BX, [EBP+18]

- a) Es incorrecta, porque el primer operando debiera ser un registro de 32 bits.
- b) Es incorrecta, porque el primer operando debiera ser un registro de 8 bits.
- c) Es correcta sintácticamente y se usa SS en el cálculo de la dirección física.
- d) Es correcta sintácticamente y se usa DS en el cálculo de la dirección física.

2) La siguiente secuencia de instrucciones:

```
TEST AL, 8  
JZ etiq
```

.....

etiq:

- a) Salta a etiq si AL=8.
- b) Salta a etiq si el bit 3 de AL es 0.
- c) Nunca salta a etiq, porque TEST hace un AND y JZ analiza los flags como si provinieran de una comparación.
- d) Es incorrecta sintácticamente y daría error al ensamblar.

3) Sabiendo que el micro está en modo real, que IDTR contiene: Base=0 y Límite=3FFH, y que la rutina de atención a la interrupción 0F3H comienza en la dirección 1234H:5678H, selecciona la afirmación correcta:

- a) La dirección de memoria 3CDH contendrá el valor 12H.
- b) La dirección de memoria 3CDH contendrá el valor 34H.
- c) La dirección de memoria 3CDH contendrá el valor 56H.
- d) La dirección de memoria 3CDH contendrá el valor 78H.



Problema 2 parcial. (Febrero tarde).

(Cont).

- 4) En un 8086, la dirección de la rutina de atención a la interrupción de vector 9 está en las siguientes direcciones de memoria:
- a) 0036H y siguientes
 - b) 0012H y siguientes
 - c) 0024H y siguientes
 - d) 0039H y siguientes
- 5) Sobre el flag de dirección (F_D) podemos decir que...
- a) ... sólo interviene cuando el microprocesador trabaja en modo protegido.
 - b) ... sólo interviene en las operaciones de manejo de cadenas (*strings*).
 - c) ... interviene en las operaciones aritméticas de multiplicar y dividir.
 - d) ... sólo interviene como complemento al flag de acarreo.
- 6) Si después de ejecutar la macro `FIND_FIRST_FILE` el programa devuelve $F_C = 0$:
- a) Significa que no se ha encontrado fichero coincidente
 - b) Se ha encontrado fichero coincidente y sus datos están en el DTA
 - c) Se ha encontrado el fichero pero no se encuentra el DTA
 - d) El path es incorrecto
- 7) El registro `IDTR`...
- a) ... no interviene en modo real
 - b) ... permite acceder mas rápidamente al modo protegido
 - c) ... indica la base y el límite de la tabla de vectores de interrupción
 - d) ... sólo actúa cuando el micro trabaje en multitarea



Problema 2 extraordinaria. (Febrero tarde).

(3 Puntos).

En un procesador IA-32 que opera en modo protegido con paginación desactivada, se está ejecutando una tarea de 16 bits con CPL=3. En un determinado momento, esa tarea deberá utilizar una rutina, propia del sistema operativo, que se encuentra en un segmento de código, que tiene un tamaño de A00H bytes, cuyo descriptor está almacenado en la posición de índice 200H de la GDT. Se dispone, además, de los siguientes datos de las tablas de descriptores del procesador:

Tabla	Dirección Base	Límite
GDT	1234CC00H	456FH
LDT	03427000H	1FFH
IDT	0023B000H	9FFH

1. ¿Qué direcciones físicas ocupa el descriptor del segmento en el que se encuentra la rutina? (0,4 ptos.)
2. El segmento de código mencionado se encuentra almacenado en la dirección física 53BD3700H. Si la rutina indicada debe acceder a zonas de datos privilegiadas y se encuentra situada en el offset 890H del segmento, obtenga justificadamente el descriptor del segmento de código, teniendo en cuenta todos los datos proporcionados por el enunciado. **Si para alguno de los campos no dispone de la información necesaria, indíquelo y considérela cero.** (0,8 ptos.)
3. Sabiendo que el segmento contiene más de una rutina ¿Podría la tarea realizar una **llamada directa** a alguna de ellas? Justifique la respuesta. (0,6 ptos.)

Suponiendo que la rutina se usa para atender una interrupción software de vector 2CH.

4. ¿Cuál sería el descriptor de *gate* más adecuado para esta situación? Obtenga justificadamente el valor de todos sus campos. **Si para alguno de los campos no dispone de la información necesaria, indíquelo y considérela cero.** (0,8 ptos.)
5. ¿Qué direcciones físicas ocuparía ese descriptor de *gate*? (0,4 ptos.)

SOLUCIÓN

1. Dado que el descriptor se encuentra en la tabla GDT, el rango de direcciones que ocupa será:

$$\text{Dirección Base} + \text{Índice} * 8 = 1234CC00H + 200H * 8 = 1234DC00H \text{ (Dirección inicial)}$$
$$1234DC07H \text{ (Dirección final)}$$



Problema 2 extraordinaria. (Febrero tarde).

(Cont).

2.

CAMPO	VALOR	SIGNIFICADO
Dirección Base	53BD3700H	Dirección física de comienzo del segmento, proporcionada por el enunciado.
Límite	9FFH	El enunciado dice que el tamaño es A00H, el límite se obtendrá como el (tamaño-1).
Gr.	0	Puesto que el tamaño proporcionado no es múltiplo de 4KB.
D	0	Tarea de 16 bits con offsets de 16 bits.
Av	0	No hay datos al respecto.
P	1	El segmento está presente en la memoria.
DPL	0	Pertenece al sistema operativo.
Tipo	11	Descriptor de segmento de código.
C	0	Código no conforme para poder acceder a zonas de datos privilegiadas.
R	0	No hay datos al respecto.
A	0	No hay datos al respecto.

De donde el descriptor resulta: 53 00 98 BD 37 00 09 FF H

3. Usando el descriptor de segmento anterior, sería imposible realizar una llamada directa a cualquier rutina incluida en ese código, puesto que está calificado como no conforme de DPL=0 y la tarea tiene CPL=3.



Problema 2 extraordinaria. (Febrero tarde).

(Cont).

4.

CAMPO	VALOR	SIGNIFICADO
Offset	890H	Offset de acceso a la rutina en cuestión dentro del segmento.
Selector	1000H	Índice=200H, TI=0 (GDT), RPL=0 (vacío).
P	1	Condición necesaria para cualquier descriptor.
Gate DPL	11	Para que sea accesible a una tarea de CPL=3.
Tipo y bit 16/32	0011(1/0)	Descriptor Interrupt o Trap Gate de 16 bits (tarea de 16 bits con offsets de 16 bits).
Byte 4	0	Una rutina de atención a interrupción no admite parámetros por pila.

Al tratarse de una interrupción software, se pueden usar ambos tipos de descriptor, aunque el tipo Trap Gate puede ser, en la mayoría de los casos, el más adecuado. Con esto, el descriptor resulta: 00 00 E(6/7) 00 10 00 08 90 H

5.

Dado que se trata de una rutina de atención a interrupción, el descriptor debe situarse en la tabla IDT, el rango de direcciones que ocupa será:

$$\text{Dirección Base IDT} + \text{vector} * 8 = 23B000H + 2CH * 8 = 23B160H \text{ (Dirección inicial)}$$
$$23B167H \text{ (Dirección final)}$$



Problema 3 extraordinaria. (Febrero tarde).

(2 Puntos).

En un sistema basado en un microprocesador IA-32 con la paginación activada, una tarea intenta realizar un acceso de escritura en memoria a una DWORD, y como la traducción de la página lineal a la que pertenece a página física no está en ese momento en el TLB, se producen los siguientes accesos en el orden que se indican:

TIPO DE ACCESO	TAMAÑO DEL DATO	DIRECCIONES
Lectura	Doble Word	6B759C10H a 6B759C13H
Lectura	Doble Word	6B759C14H a 6B759C17H
Lectura	Doble Word	75BC18588H a 75BC1858BH
Lectura	Doble Word	75BC1858CH a 75BC1858FH
Lectura	Doble Word	39F200720H a 39F200723H
Lectura	Doble Word	39F200724H a 39F200727H
Escritura	Doble Word	B596C5D28H a B596C5D2BH

Además se dispone del siguiente volcado de memoria:

DIRECCIÓN	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
06B759C00H	3F	36	92	1C	0B	00	00	00	86	52	65	0E	06	00	00	00
06B759C10H	19	8E	CI	5B	07	00	00	00	52	3A	08	76	0C	00	00	00
39F200710H	3A	B2	00	00	0F	00	00	00	58	F7	9C	49	0D	00	00	00
39F200720H	15	55	6C	59	0B	00	00	00	19	CA	3A	76	04	00	00	00
75BC18580H	68	DB	00	00	00	00	00	00	35	0A	20	9F	03	00	00	00
75BC18590H	24	02	C4	43	92	00	00	00	90	8F	3A	7A	1F	00	00	00

Se pide (responda en los recuadros destinados al efecto, usando el resto del espacio disponible para las operaciones necesarias para desarrollar el ejercicio):

- 1) Indique razonadamente el valor del bit PAE y el tamaño de página en Bytes de este acceso (0.4ptos.).

PAE= 1
Tamaño de la página: 4KB

Justificación: *Se obtienen tres datos de 8 Bytes de tres posiciones distintas de memoria, eso sólo puede ocurrir en máquinas con direccionamiento extendido a página de 4KB.*



Problema 3 extraordinaria. (Febrero tarde).

(Cont).

2) Contenido de las entradas de las tablas involucradas en el acceso (0.4ptos.).

TABLA	DIRECCIÓN	CONTENIDO
PDPT	06B759C10H	000000075BC18E19H
PD	75BC18588H	000000039F200A35H
PT	39F200720H	0000000B596C5515H

3) Dirección lineal a la que se intenta acceder (0.4ptos.).

Dirección lineal: 962E4D28H

Se obtiene al tener en cuenta los cuatro campos en los que se divide la dirección lineal, dado el tipo de acceso de que se trata.

4) ¿Qué datos se guardan en el TLB después de este acceso? (0.4ptos.).

VALOR	CONTENIDO
Nº de página lineal	962E4H
Nº de página física	B596C5H
U/Se (1)	1
R/We (1)	0
G (2)	1
D (3)	1

(1) Observando los bits de control de las entradas de PD y PT y tomando sus valores efectivos.

(2) id. de PT.

(3) id. de PT y tras el acceso.



Problema 3 extraordinaria. (Febrero tarde).

(Cont).

- 5) Si antes de que se borren estos datos del TLB se intenta acceder de nuevo a la misma página, indique en función del tipo de acceso (lectura, escritura o fetch) y del CPL en curso, cual de estas opciones ocurriría. (0.4ptos.):
- Realiza la traducción a través de los datos del TLB y no accede ni al PD ni a la PT.
 - Necesita realizar un acceso a la PT, pero no tiene porqué pasar por el PD.
 - Se produce una excepción, y por lo tanto no se llega a realizar el acceso.
 - El acceso requiere leer y/o escribir previamente las entradas del PD y de la PT.

-CPL < 3 (modo supervisor): Independientemente del tipo de acceso, la opción es la a), pues no hay restricciones y ya se ha realizado acceso de escritura.

-CPL = 3 (modo usuario): Tiene restringido el acceso de escritura, luego se produciría lo que indica la opción c). Para el resto de modos de acceso vuelve a suceder la opción a)

Depto. de Electrónica y Comunicaciones de la U.P.S.



Problema 4 extraordinaria. (Febrero tarde).

(1 Puntos).

En las siguientes preguntas de tipo test, sólo existe una respuesta correcta. Cada pregunta tiene un valor de 0,2 pts. Cada respuesta incorrecta descuenta 0,05 pts. La pregunta que esté en blanco no sumará ni restará puntos.

Notas: • Se debe poner una X sobre la letra de la tabla "SOLUCIÓN" (que encontrará en la última hoja) correspondiente a la respuesta correcta.

• La tabla "CALIFICACIÓN" no deberá utilizarla. Es para su posterior calificación.

• No se tendrán en cuenta las respuestas que no se encuentren en dicha tabla "SOLUCIÓN".

1) ¿Cuál de las siguientes afirmaciones relacionadas con el protocolo MESI es cierta?.

- a) Cuando se descarga una línea de caché sobre la memoria principal, se hace simultáneamente un sondeo para determinar si está presente en otras cachés.
- b) Cuando se carga una línea en caché, se hace simultáneamente un sondeo para determinar si está presente en otras cachés.
- c) El funcionamiento de este protocolo no requiere ningún tipo de sondeo.
- d) Cuando una CPU modifica una línea que está en su caché en estado *S*, se envía por el bus la modificación de la línea para que todas las cachés que la tengan, la actualicen.

2) ¿Cuál de las siguientes afirmaciones relacionadas con los criterios de escritura es cierta?.

- a) El criterio *write back* requiere un bit más de status que el *write through*.
- b) El criterio *write through* requiere, al menos, 2 bits de status.
- c) El criterio *posted write through* requiere, al menos, 2 bits de status.
- d) El número de escrituras en memoria principal que se realiza es independiente del criterio de escritura.

3) En un sistema de caché se sabe que el tamaño máximo de la memoria principal es de 8GB, la caché de líneas (o de datos) es de 1 MB, la memoria de etiquetas es de 32K17 y el protocolo que usa es el MESI. ¿Cuál es el tamaño de la memoria de algoritmo sabiendo que usa el LRU?

- a) 32Kb.
- b) 16K3.
- c) 8K6.
- d) Ninguno de los anteriores.



Problema 4 extraordinaria. (Febrero tarde).

(Cont).

- 4) En un sistema multiprocesador que usa el protocolo MESI, un procesador realiza un acceso a una línea que tiene marcada como S. Si esa línea cambia de estado, el nuevo estado podrá ser:
- a) M.
 - b) E.
 - c) S.
 - d) I.
- 5) En un sistema monoprocesador conectado con una caché directamente “mapeada”, se sabe que: $i=4$, $j=8$ y $k=16$. Si el micro intenta acceder a la dirección 593CD17H. ¿Cuál es el valor del campo C2 (etiqueta)?.
- a) 17H.
 - b) CD17H.
 - c) 593CH.
 - d) 483AH.

SOLUCIÓN

1	2	3	4	5
a	a	a	a	a
b	b	b	b	b
c	c	c	c	c
d	d	d	d	d

**Problema 1 final. (Junio mañana).****(4 Puntos).**

Dado el siguiente programa:

```
INCLUDE macros.mac
MODEL SMALL

MACRO Mac1

STACK 200H
DATASEG
var1 DB ?
var2 DB ?
var3 DB 80 DUP (?)
var4 DB 21 DUP (?)
var5 DB ?
var6 DB 4 DUP (?)
var7 DD ?
var8 DB 13 DUP (?)
var9 DD 0
var10 DD 0
var11 DD 0
var12 DD 0
Msg1 ...
Msg2 ...
Msg3 ...
Msg4 ...
Msg5 DB ' Bytes',13, 10, 0
Msg6 ...
Msg7 ...
Err1 ...
Err2 ...
cr_lf DB 13, 10, 0
Tabla DB '0123456789'

CODESEG
P386N
EXTRN UDATA:NEAR
STARTUPCODE
DISP_STRINGZ Msg1
GET_STRING 80, var1
DISP_STRINGZ cr_lf
XOR BX, BX
MOV BL, var2
MOV var3[BX], 0
SET_DTA var4
FIND_FIRST_FILE var3, 7
JC Error1

eti1: INC var10
MOV EBX, var7
ADD var9, EBX
MOV AL, var5
AND AL, 00100000b
JZ eti2
INC var12
ADD var11, EBX
eti2: FIND_NEXT_FILE
JNC eti1
eti3: DISP_STRINGZ Msg2
READ_KBD_AND_ECHO
PUSH AX
DISP_STRINGZ cr_lf
POP AX
CMP AL, 'B'
JE eti5
CMP AL, 'K'
JE eti4
CMP AL, 'M'
JNE eti3
CALL rut1
eti4: CALL rut1
MOV Msg5[1], AL
eti5: DISP_STRINGZ Msg3
Mac1 var10
DISP_STRINGZ cr_lf
DISP_STRINGZ Msg4
Mac1 var9
DISP_STRINGZ Msg5
DISP_STRINGZ cr_lf
DISP_STRINGZ Msg6
Mac1 var12
DISP_STRINGZ cr_lf
DISP_STRINGZ Msg7
Mac1 var11
DISP_STRINGZ Msg5
DISP_STRINGZ cr_lf
END_PROCESS 0

Rutina rut1

Error1:
.....
end
```

NOTA: En la variable *Msg5* hay 2 espacios en blanco delante de la palabra “Bytes”.



Problema 1 final. (Junio mañana).

(Cont).

Se pide:

- A. Codificar la macro *MAC1* para que realice lo siguiente:
Recibirá como parámetro un entero sin signo de 32 bits. Convertirá el número a cadena ASCIIZ (se puede usar *var3* como variable para contener la cadena convertida) en base 10 y con un tamaño mínimo de un carácter. A continuación mostrará dicha cadena por pantalla. (0,4 ptos.)
- B. Codificar *rut1* para que realicen lo siguiente:
Divide *var9* entre 1024. El cociente se almacena en *var9* y si el resto de la división es mayor de 511, se incrementará el cociente en una unidad. Obsérvese que 1024 es potencia de 2. Repetir el mismo proceso con *var11*. (0,4 ptos.)
- C. Analiza el programa en su conjunto. Una vez que tengas completamente claro su funcionamiento indica en líneas generales que es lo que hace. Se debe explicar al menos: ¿Qué entradas de datos se le piden al usuario? ¿Qué proceso se lleva a cabo con los datos introducidos por el usuario? ¿Qué muestra el programa como resultado de su ejecución? (1 pto.)
- D. Indica qué información se almacena en cada una de las variables desde *var1* hasta *var12*. (1 pto.)
- E. Define correctamente y con textos significativos las variables *Msg1* a *Msg7*, excepto *Msg5* que ya esta correctamente declarada. (1 pto.)
- F. Escribe el código que debe aparecer junto a la etiqueta de tratamiento de errores *Error1*. Si existen diferentes causas para este error, se mostrará un mensaje específico (*Err1*, *Err2*...) para cada una de esas causas, y en todos los casos se finalizará el programa con un código de retorno igual a 1. (0,2 ptos.)

Nota: Cada apartado debe resolverse en las zonas indicadas. Para anotaciones en sucio pueden usarse los reversos de las hojas de este problema.

SOLUCIÓN

APARTADO A)

```
Mac1    MACRO num
PUSH    DWORD PTR num
PUSH    WORD PTR 10
PUSH    WORD PTR 1
PUSH    OFFSET Tabla
PUSH    OFFSET var3
CALL    UDTA
DISP_STRINGZ var3
ENDM
```



Problema 1 final. (Junio mañana).

(Cont).

APARTADO B)

```
rut1  PROC NEAR
      SHR      var9, 10
      ADC      var9, 0
      SHR      var11, 10
      ADC      var11, 0
      RET
rut1  ENDP
```

APARTADO C)

*El programa solicita que el usuario teclee un pathname que puede incluir los comodines * e ?. A partir de esa información se buscarán todos los ficheros que se correspondan con el mismo y calculará cuántos ficheros son y cuántos bytes ocupan. Al finalizar la búsqueda si no se encuentra ningún fichero se muestra un mensaje relativo a esa situación, Asimismo si el pathname es léxicamente incorrecto se muestra un error y si no se dan estos casos, se solicita al usuario que teclee los caracteres 'B?', 'K' o 'M?' según quiera visualizar la información en Bytes, KBytes o MBytes. Si se teclea un carácter incorrecto se vuelve a solicitar. Finalmente se muestra el número de ficheros encontrados y el espacio que ocupan y se muestra otra vez la misma información pero en este caso de los ficheros 'modificados' después de la última copia de seguridad.*



Problema 1 final. (Junio mañana).

(Cont).

APARTADO D)

var1, var2 y var3 Se corresponden con la estructura que usa la Función GET_STRING.

Concretamente var1 es el máx número de caracteres de la cadena a introducir incluyendo el CR. var2 contendrá (al retornar de la función) el número de caracteres realmente teclado sin CR y var3 la cadena tecleada por el usuario.

var4 a var8 se utilizan como DTA, es decir contendrán los 43 bytes que entrega el S.O. después de encontrar un fichero con las funciones FIND_FIRST_FILE o FIND_NEXT_FILE. Concretamente var4 contendrá la parte reservada del DTA, var5 contiene el byte de atributos, var6 tendrá la fecha y la hora, var7 el tamaño del fichero y var8 el nombre del fichero encontrado.

var9 es un acumulador de los bytes ocupados por todos los ficheros encontrados.

var11 tiene la misma función pero de los ficheros que han sido modificados desde la última copia de seguridad.

var10 es un contador del número total de ficheros encontrados y var12 hace lo propio pero sólo de los ficheros modificados desde la última copia de seguridad.

APARTADO E)

```
Msg1  DB      'Introduce Pathname de los ficheros a buscar: ', 0
Msg2  DB      'Introduce "B", "K" o "M" según desees Bytes, Kbytes o MBytes: ', 0
Msg3  DB      "El nº total de ficheros encontrados es: ",0
Msg4  DB      "Los ficheros encontrados ocupan: ",0
Msg5  DB      ' Bytes',13, 10, 0
Msg6  DB      "El nº de ficheros modificados es de : ",0
Msg7  DB      "Los ficheros modificados ocupan: ",0
```

APARTADO F)

```
Error1:  CMP     AX, 2
         JE     PathNoVal
         DISP_STRINGZ Err1
         END_PROCESS 1
PathNoVal: DISP_STRINGZ Err2
         END_PROCESS 1

Err1     DB      'Ningún fichero encontrdo', 13, 10, 0
Err2     DB      'Path name no válido', 13, 10, 0
```



Problema 2 final/1 parcial. (Junio mañana).

(3/5 Puntos).

Un procesador IA-32, operando en modo protegido con paginación desactivada, ejecuta la instrucción **CALL 122FH:0000**. Durante la ejecución se escribe en el intervalo B527A304H a B527A315H de memoria, con lo que el contenido de esa zona queda como aparece en el siguiente volcado:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
B527A300H	00	00	00	00	D0	02	C7	03	55	37	2A	B3	00	16	0D	44
B527A310H	99	38	0C	FF	2F	11	33	A1	00	12	92	4B	81	C6	11	00

Además, se dispone de los siguientes datos relativos a las tablas de descriptores del microprocesador:

Tabla	Dirección Base	Límite
LDT	2000000H	1DFFH
GDT	3000000H	CFFFH
IDT	5000000H	177H

Y del siguiente volcado de memoria:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20003B0H	3A	B2	00	00	0F	00	00	00	58	F7	9C	49	0D	00	00	00
20003C0H	B0	0A	00	D0	2C	FF	00	35	19	CA	3A	76	04	00	00	00
2001220H	3F	36	92	1C	0B	00	00	00	C0	05	98	0D	05	E4	00	00
2001230H	27	B4	A9	62	08	00	00	00	52	3A	08	76	0C	00	00	00
3000D80H	68	DB	00	00	00	EF	CD	AB	00	00	FF	B7	A2	87	4C	D7
3000D90H	24	02	C4	43	92	D4	00	C7	02	00	00	A0	12	B9	80	13

Teniendo en cuenta todos los datos anteriores, responda a los siguientes apartados.

- Indique razonadamente qué tipo de proceso se ha llevado a cabo al ejecutar esa llamada a rutina. ¿Ha habido cambio de CPL? Si ha sido así, ¿qué consecuencias ha tenido?. (0,3/0,5 ptos.)
- Obtenga el o los descriptores que se han usado durante la ejecución de la instrucción CALL anterior. Descompongalos especificando el significado de cada uno de sus campos. (0,6/1,0 ptos.)
- ¿Qué valores tendrán los registros CS (parte visible y parte invisible) y (E)IP inmediatamente después de ejecutar la instrucción CALL? (0,6/1,0 ptos.)
- Suponiendo que la rutina retorna al programa llamador, cuál sería la instrucción de retorno más adecuada (detalle lo más posible esa instrucción). Obtenga el descriptor de segmento en el que se encuentra el código llamador, especificando el significado de cada uno de sus campos. (0,6/1,0 ptos.)
- Si se pretende usar la rutina como respuesta a la interrupción software 2AH, obtenga el descriptor que debería generar el S.O. para que fuera posible. ¿En qué rango de direcciones de memoria habría que situar ese descriptor? ¿Debería modificarse de alguna manera la rutina para que se pueda vincular a esa interrupción? ¿Cuál es el mayor valor de vector utilizable en el sistema?. (0,9/1,5 ptos.)



Problema 2 final/1 parcial. (Junio mañana).

(Cont).

SOLUCIÓN

1)

Interpretando la escritura realizada en la pila, se deduce que se ha realizado una carga indirecta de CS con cambio de CPL con cambio de pila y transferencia de parámetros. Donde, las WORDs escritas corresponden a (en orden de B527A304H a B527A315H) IP, CS_{PV}, parámetros (5 WORDs), SP, SS_{PV}.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
B527A300H	00	00	00	00	D0	02	C7	03	55	37	2A	B3	00	16	0D	44
B527A310H	99	38	0C	FF	2F	11	33	A1	00	12	92	4B	81	C6	11	00

Diagram showing labels for IP (bits 4-5), CS_{PV} (bits 6-7), SP (bits 2-3), and SS_{PV} (bits 10-11).

2)

Se han usado dos, uno de CALL GATE y otro de segmento de código.

Descriptor CALL GATE: A partir de la instrucción CALL 122FH se accede al rango de posiciones: 2001228H a 200122FH, de donde se lee el descriptor.

7	6	5	4	3	2	1	0
00	00	E4	05	0D	98	05	C0
Offset 31-16		D.A.	WC	Selector		Offset 15-0	

7	6	5	4	3	2	1	0
1	1	1	0	0	1	0	0
P	DPL		16/3				

Descriptor de segmento de código: A partir del selector que aparece en el descriptor CALL GATE se accede a GDT en las posiciones: 3000D98H a 3000D9FH, donde se encuentra el descriptor:

7	6	5	4	3	2	1	0
13	80	B9	12	A0	00	00	02
D.B.35-24		D.A.	Dirección Base 24-0			Límite 15-0	

7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0
Gr	D	0	Av		Límite 19-16		

7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	1
P	DPL		Tipo		C	R	A

Se trata de un descriptor de segmento de código no conforme, no legible, con privilegio 1, de 3000H Bytes que comienza en la dirección 1312A000H.



Problema 2 final/1 parcial. (Junio mañana).

(Cont).

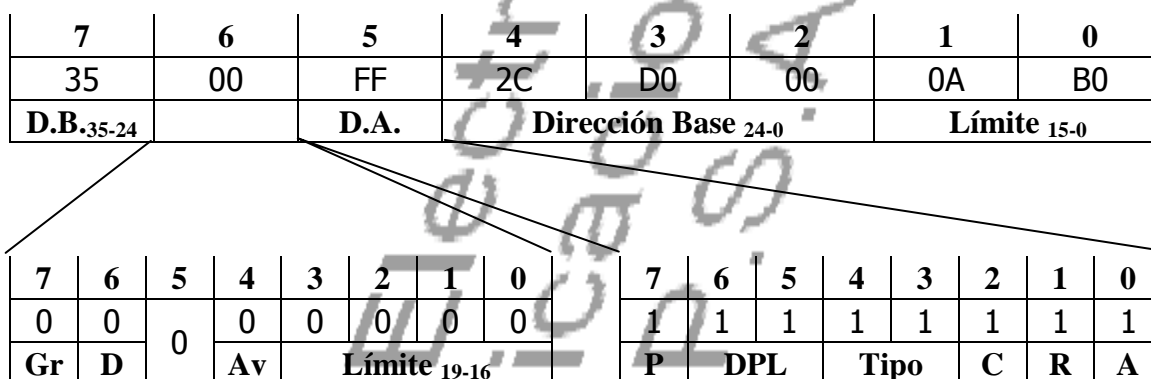
3)

CS _{PV}	0D99H
CS _{PI}	1380B912A0000002H
IP	05C0H

4)

La instrucción más adecuada es RET 10. Para que la rutina descargue los parámetros de la pila.

Teniendo en cuenta el selector cargado en pila al ejecutar la instrucción CALL: 03C7H. El descriptor se encuentra en el rango de direcciones: 20003C0H a 20003C7H.

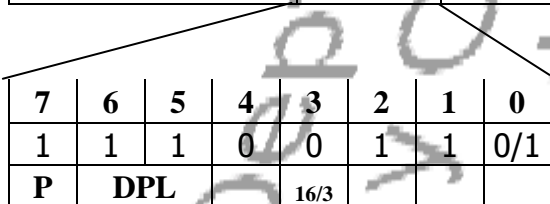


Se trata de un descriptor de segmento de código conforme, legible, con privilegio 3, de AB1H Bytes que comienza en la dirección 352CD000H.

5)

Debería tratarse de un descriptor INTERRUPT o TRAP GATE, con características similares al de CALL GATE del apartado 1) salvo por el número de parámetros (WC).

7	6	5	4	3	2	1	0
00	00	E6/7	00	0D	98	05	C0
Offset₃₁₋₁₆		D.A.	----	Selector		Offset₁₅₋₀	



El rango de direcciones en el que habría que colocar el descriptor sería 5000150H a 5000157H, dentro de la IDT.

La rutina no debería esperar parámetros por pila y tendría que finalizar con un IRET.

Teniendo en cuenta que el límite de IDT es 177H el vector más alto admitido será 2EH.



Problema 3 final^{1/2} parcial. (Junio mañana).

(2/3 Puntos).

De una configuración multiprocesador como la esquematizada en la figura, se sabe lo siguiente:

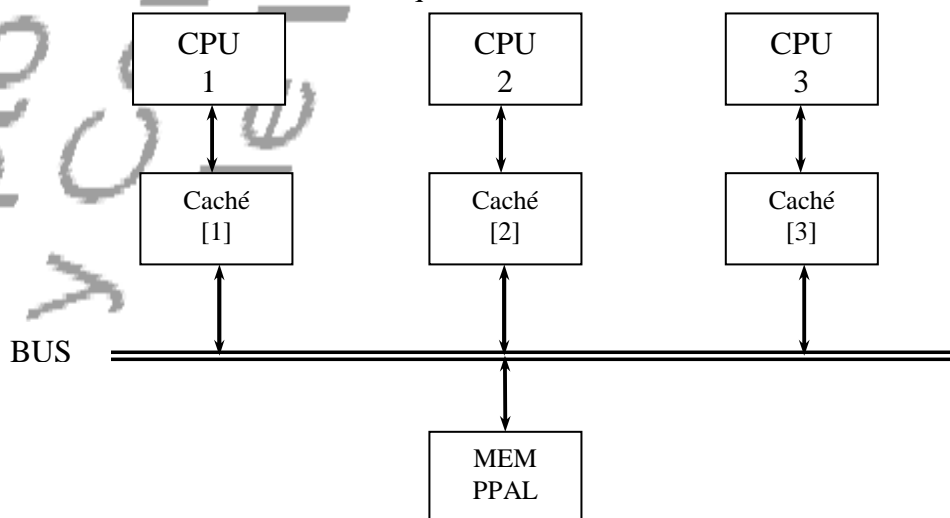
- A) Las direcciones físicas de memoria se codifican en 30 bits.
- B)

	Capacidad memoria de líneas	Posiciones de memoria de etiquetas	Criterio de escritura
Caché [1]	1 MB	¿?	Write back
Caché [2]	64 KB	2 K	Write through
Caché [3]	512 KB	¿?	Write back

- C) Cada posición de la memoria de etiquetas de la caché [1] tiene 13 bits, en los que se incluyen los bits que codifican el estado de la línea.
- D) La caché [2] está *directamente mapeada*.
- E) La memoria de algoritmo de la caché [3] tiene 4K posiciones.
- F) Las memorias caché que usan el criterio de escritura *write back* (escritura obligada) siguen el protocolo MESI y la de criterio *write through* un bit de *status*.
- G) Todas las memorias caché codifican el estado de la línea con los bits **más** significativos de la memoria de etiquetas según la siguiente tabla:

MESI		STATUS DE 1 BIT	
ESTADO	CÓDIGO	ESTADO	CÓDIGO
M	11	Válido	1
E	10	Inválido	0
S	01		
I	00		

- H) Se sabe que en un determinado momento, la línea que comienza en la dirección de memoria principal 1543AB60H se encuentra también almacenada en las tres memorias caché, que sendas copias están actualizadas, y que cada una está situada en el último plano (o vía) de la caché correspondiente.



¹ En el examen final se eliminaron los apartados 2 y 6.



Problema 3 final/2 parcial. (Junio mañana).

(Cont).

Se pide:

1). Rellene la siguiente tabla:

(0,5/0,6 ptos.)

	i	j	k	n
Caché [1]	5	14	11	2
Caché [2]	5	11	14	1
Caché [3]	5	12	13	4

2). Tamaño (número de posiciones y número de bits por posición) de las memorias de etiquetas de las cachés [1], [2] y [3].

(0,0/0,5 ptos.)

Caché[1]: 32K posiciones de (11+2) bits/posición = 32K13

Caché[2]: 2K posiciones de (14+1) bits/posición = 2K15

Caché[3]: 16K posiciones de (13+2) bits/posición = 16K15

3). ¿Cuál es la dirección de caché equivalente a la de memoria principal especificada en el último apartado para las tres memorias?.

(0,5/0,5 ptos.)

Caché[1]	n_1	j_1	i_1	Dir=BAB60H
	1	011 1010 1011 011	00000	
Caché[2]	n_1	j_1	i_1	Dir=AB60H
	-	1010 1011 011	00000	
Caché[3]	n_1	j_1	i_1	Dir=7AB60H
	11	1 1010 1011 011	00000	

4). ¿Cuál es el contenido de la memoria de etiquetas correspondiente a la posición citada para las tres cachés?.

(0,5/0,5 ptos.)

Caché[1]	status	etiqueta	Contenido= 0AA8H
	01	01 0101 0100 0	
Caché[2]	status	etiqueta	Contenido= 5543H
	1	01 0101 0100 0011	
Caché[3]	status	etiqueta	Contenido= 2AA1H
	01	01 0101 0100 001	



Problema 3 final/2 parcial. (Junio mañana).

(Cont).

- 5). La CPU 3 realiza una operación de escritura sobre esa línea. Indicar que acciones se realizan sobre el bus, la memoria principal, la memoria de datos, y la memoria de etiquetas por parte de cada una de las cachés. Si alguna etiqueta cambia su contenido debe indicarse el nuevo valor.

(0,5/0,6 ptos.)

	Caché 1	Caché 2	Caché 3
Acciones sobre el bus	---	---	Envía orden de invalidar línea (o BusRdX, según la implementación) (*)
Acciones sobre la memoria principal	---	---	---
Acciones sobre la caché de datos	---	---	Se permite la escritura del procesador directamente
Acciones sobre la caché de etiquetas	Cambia su estado a I (inválido)	Cambia su estado a I (inválido)	Cambia su estado a M (modificado)
Nuevo valor de la etiqueta (si cambia)	2A8H	1543H	6AA1H

(*) Si en el sistema no existiera específicamente la orden del bus "invalidar línea", se habría lanzado una orden BusRdX y, en contestación, la memoria principal hubiera enviado la línea, aunque la caché 3 no la recogería porque no le hace falta.

- 6). ¿Qué operaciones se hubieran producido por el bus si en el apartado anterior el acceso de la CPU3 hubiera sido en lectura en vez de en escritura?.

(0,0/0,3 ptos.)

	Caché 1	Caché 2	Caché 3
Acciones sobre el bus	---	---	No se hubiera realizado ninguna acción



Problema 4 final²/3 parcial. (Junio mañana).

(1/2 Puntos).

En las siguientes preguntas de tipo test, sólo existe una respuesta correcta. Cada pregunta tiene un valor de 0,2 pts. Cada respuesta incorrecta descuenta 0,05 pts. La pregunta que esté en blanco no sumará ni restará puntos.

Notas: • Se debe poner una X sobre la letra de la tabla "SOLUCIÓN" (que encontrará en la última hoja) correspondiente a la respuesta correcta.

• La tabla "CALIFICACIÓN" no deberá utilizarla. Es para su posterior calificación.

• No se tendrán en cuenta las respuestas que no se encuentren en dicha tabla "SOLUCIÓN".

1) En la paginación puede afirmarse que...

- a) ... en un mismo sistema no pueden convivir en ningún caso páginas de 4MB con páginas de 2MB.
- b) ... en un mismo sistema no pueden convivir en ningún caso páginas de 4KB con páginas de 4MB.
- c) ... en un mismo sistema no pueden convivir en ningún caso páginas de 4KB con páginas de 2MB.
- d) Ninguna de las anteriores es cierta.

2) En la paginación puede afirmarse que...

- a) ... si en la entrada de la tabla correspondiente el bit G está a uno, lo que indica es que la página apuntada tiene datos más importantes que otras páginas con $G = 0$.
- b) ... los bits U/S y W/R imponen restricciones a las tareas con $CPL > 0$.
- c) ... el rendimiento del sistema (en velocidad) se ve afectado negativamente.
- d) ... en todos los casos las entradas de PDs apuntan a PTs.

3) Una tarea con $CPL=1$ intenta acceder a una WORD para escritura, y la entrada de la última tabla de paginación contiene el valor 3CCCA15FH ...

- a) ... se realiza la escritura correctamente, y se modifica dicha entrada.
- b) ... se realiza la escritura correctamente, pero no se modifica dicha entrada.
- c) ... produce una excepción.
- d) ... el valor indicado no es válido para la última tabla de paginación.

² En el examen final había sólo cinco de estas diez preguntas (se eliminaron las cuestiones 2, 4, 6, 8 y 10).



Problema 4 final/3 parcial. (Junio mañana).

(Cont).

- 4) Sabiendo que se accede a la dirección física 312345678H se puede afirmar que la dirección lineal puede ser:
- a) A58BCA78H.
 - b) A5345678H y la página es de 2 MB.
 - c) 12345678H y la página es de 4 MB.
 - d) Esa dirección física no es válida.
- 5) Una tarea con CPL=1 intenta acceder a una WORD para escritura, y la entrada de la última tabla de paginación contiene el valor 8A945F7CH ...
- a) ... produce una excepción.
 - b) ... se realiza la escritura correctamente (sin que se produzca excepción alguna) sin modificar dicha entrada.
 - c) ... se realiza la escritura correctamente, pero se modifica dicha entrada.
 - d) ... el valor indicado no es válido para una tabla de paginación.
- 6) El espacio de direccionamiento lineal para una tarea ...
- a) ...en modo extendido es mayor que en modo no extendido.
 - b) ...en modo extendido es igual que en modo no extendido.
 - c) ...en modo extendido es menor que en modo no extendido.
 - d) Ninguna de las anteriores es correcta.
- 7) En un microprocesador con arquitectura IA-32 si se accede a la dirección 123456H ...
- a) ...se sabe que se está trabajando en modo extendido.
 - b) ...se sabe que se está trabajando en modo no extendido.
 - c) ...no se puede asegurar que estamos en modo extendido ni en modo no extendido.
 - d) Ninguna de las anteriores es correcta.



Problema 1 final. (Junio tarde).

(4 Puntos).

Dado el programa incompleto que aparece a continuación:

- 1) Diseñe el **FRAGMENTO1**, que debe sacar por pantalla el mensaje **msg1**, solicitando que se teclee una secuencia de, como máximo, 16 caracteres, dejándola almacenada en **var3**, y, a continuación, convertirla en ASCII. Indique también, de la forma más explícita posible, el contenido del mensaje **msg1**. (0,5 pts)
- 2) Comente brevemente la funcionalidad del fragmento señalado con el **comentario a**. (0,5 pts)
- 3) Diseñe el **FRAGMENTO2**, para que realice lo siguiente:
 - a) Llamar a la rutina **buscar**; pasándole los 4 parámetros de tamaño WORD que requiere, que son:
 - i) 1^{er} parámetro: número de bytes obtenidos por la System Call **READ_HANDLE** que aparece en el fragmento identificado mediante el **comentario a**.
 - ii) 2^o parámetro: tamaño en bytes de la secuencia obtenida mediante el **FRAGMENTO1**.
 - iii) 3^{er} parámetro: offset del buffer en el que se almacenan los bytes leídos del fichero del apartado i).
 - iv) 4^o parámetro: offset de la variable **var3**.
 - b) Comprobar si la rutina ha devuelto **AX = 0** o **AX = 1**: en el primer caso, saltar a **eti2**, y, si es uno, continuar con la ejecución del fragmento identificado por el **comentario b**.
Nótese que el código de la rutina **buscar**, ni se proporciona ni se pide, pero se sabe que lo que hace es localizar una cadena, identificada por su offset inicial (parámetro 4) y su tamaño (parámetro 2), en un buffer, identificado también del mismo modo (offset inicial en parámetro 3 y tamaño en parámetro 1), proporcionando al final **AX = 0** si no se ha encontrado la cadena, y **AX = 1** si se ha localizado.
(0,5 pts)
- 4) Comente brevemente la funcionalidad del fragmento señalado con el **comentario b** (observe que este fragmento está dividido en dos partes en el código). Indique, de la forma más explícita posible, el contenido del mensaje **msg4**. (0,5 pts)
- 5) Comente brevemente la funcionalidad del fragmento señalado con el **comentario c**. ¿Qué mensaje aparecerá en pantalla al presentar la secuencia **msg2a**, **msg2b** y **msg2c**? (0,5 pts)
- 6) Comente la funcionalidad del fragmento señalado con el **comentario d**. Indique, de la forma más explícita posible, cuál debería ser el contenido del mensaje **msg3**. (0,5 pts)
- 7) Suponiendo que el código que hay a partir de las etiquetas **error1**, **error2**, **error3** y **error4** (este código ni se muestra ni se pide) se encarga de:
 - a) Visualizar los mensajes de error **msgerr1**, **msgerr2**, **msgerr3** y **msgerr4** respectivamente.
 - b) Cerrar los ficheros que se encuentren abiertos.
 - c) Finalizar con un parámetro de salida alusivo al error.Escriba los mensajes de error **msgerr1**, **msgerr2**, **msgerr3** y **msgerr4**, de la forma más explícita posible.
(0,5 pts)
- 8) Explique brevemente qué hace el programa en su conjunto. (0,5 pts)

NOTA: En ningún caso debe añadir ninguna variable a las que ya están definidas en el programa, usando para cada situación (apartados 1, 3 y 7) las que considere más adecuadas.

**Problema 1 final. (Junio tarde).****(Cont).**

```

INCLUDE macros.mac
MODEL SMALL
STACK 200H
DATASEG
msg1          '-----',13,10,0
msg2a        DB      'La secuencia: ',0
msg2b        DB      ' se ha localizado en ',0
msg2c        DB      ' ficheros del directorio actual.',13,10,0
msg3         DB      '-----',13,10,0
msg4         DB      '-----',13,10,0
msgerr1      DB      '-----',13,10,0
msgerr2      DB      '-----',13,10,0
msgerr3      DB      '-----',13,10,0
msgerr4      DB      '-----',13,10,0
var1         DB      ?
var2         DB      ?
var3         DB      17 DUP (?)
var4         DB      30 DUP (?)
var5         DB      13 DUP (?)
var6         DB      '*.*',0
var7         DB      'fich.log',0
var8         DB      1024 DUP (?)
var9         DD      0
var10        DW      ?
var11        DW      ?
tabla        DB      '0123456789'
cr_lf        DB      13,10,0
CODESEG

P386N
EXTRN        udta:NEAR
STARTUPCODE

FRAGMENTO1
CREATE_HANDLE var7,0 ;a
JC error1 ;a
MOV var10,AX ;a
SET_DTA var4 ;a
FIND_FIRST_FILE var6,7 ;a
JC error2 ;a
etiql1: OPEN_HANDLE var5,0 ;a
MOV var11,AX ;a
etiql2: READ_HANDLE var11,var8,1024 ;a
JC error3 ;a
CMP AX,0 ;a
JE etiq3 ;a

FRAGMENTO2
INC var9 ;b
WRITE_HANDLE var10,var5,13 ;b
CMP AX,CX ;b
JNE error4 ;b
etiql3: CLOSE_HANDLE var11 ;b
FIND_NEXT_FILE ;b
JNC etiq1 ;b
CMP var9,0 ;b

etiq4: READ_KBD
OR AL,20H ;d
CMP AL,'s' ;d
JE etiq5 ;d
CMP AL,'n' ;d
JE etiq7 ;d
DISPLAY_CHAR 7 ;d
JMP etiq4 ;d
etiq5: DISPLAY_CHAR 's' ;d
MOVE_PTR var10,0,0,0 ;d
etiq6: READ_HANDLE var10,var8,13 ;d
OR AX,AX ;d
JZ etiq8 ;d
DISP_STRINGZ var8 ;d
DISP_STRINGZ cr_lf ;d
JMP etiq6 ;d
etiq7: DISPLAY_CHAR 'N' ;d
etiq8: CLOSE_HANDLE var10 ;d
END_PROCESS 0 ;d
etiq9: DISP_STRINGZ msg4 ;b
JMP etiq8 ;b

error1: -----
error2: -----
error3: -----
error4: -----

buscar PROC NEAR
;RUTINA DE BÚSQUEDA
;(No incluida)
buscar ENDP
END

```

NOTA: Los siguientes códigos ASCII pueden ser de interés para el problema: 0 = NULL. 7 = Código de BELL (pitido de pantalla). 10 = New Line o Line Feed. 13 = Carriage Return. 41H a 5AH = Letras mayúsculas ("A" a "Z"). 61H a 7AH = Letras minúsculas ("a" a "z").

**Problema 1 final. (Junio tarde).****(Cont).****SOLUCIÓN**

1) El FRAGMENTO 1 podría quedar de la siguiente forma:

```
DISP_STRINGZ msg1
GET_STRING 17,var1
XOR BX,BX
MOV BL,c_rec
MOV var3[BX],0

msg1 DB 'Teclee la cadena a localizar.'
DB ' Máximo 16 caracteres.',13,10,0
```

2) Crea un fichero, de nombre "fich.log", en el directorio actual, si no puede crearlo se produce el error1.

Define el DTA y busca el primer fichero del directorio actual, si no hay ninguno salta a error2.

Abre el fichero encontrado e intenta leer 1KB, si no puede leer se produce el error3. Si no lee nada, pasa a etiq3.

3) El FRAGMENTO 2 se podría desarrollar de la siguiente forma:

```
PUSH AX
XOR BX,BX
MOV BL,var2
PUSH BX
PUSH OFFSET var8
PUSH OFFSET var3
CALL buscar
OR AX,AX
JE etiq2
```

4) Empieza incrementando var9, que se inicializó a 0, esto se produce si la rutina ha localizado la cadena; copia en fich.log el nombre del fichero en el que se ha localizado. Comprueba si la unidad lógica está llena, si lo está genera el error4.

Intenta buscar el siguiente fichero del directorio, si lo encuentra repite el proceso de localización de la cadena, si no comprueba si var9 es cero y si lo es saca el mensaje msg4 por pantalla y sale.

El mensaje msg4 podría definirse del siguiente modo:

```
msg4 DB 'No se ha encontrado la cadena en el directorio actual.',13,10,0
```

5) Pretende sacar por pantalla el siguiente mensaje formado por varios tramos:

"La secuencia: **<cadena a buscar>** se ha localizado en **<nº de ficheros que la contienen>** ficheros del directorio actual".



Problema 1 final. (Junio tarde).

(Cont).

6) Proporciona al usuario del programa la alternativa de imprimir en pantalla o no el listado de ficheros en los que se ha localizado la cadena buscada. El listado se encuentra en el fichero auxiliar *fich.log*, cuyo contenido se llevaría a pantalla línea a línea. El mensaje msg3 podría ser de la siguiente forma:

```
msg3 DB `¿Quiere imprimir en pantalla la lista de ficheros? (S/N)',13,10,0
```

El fragmento se encarga también de comprobar la respuesta (s ó n) del usuario, aceptando mayúsculas o minúsculas, pero ningún otro carácter.

7)

```
msgerr1 DB `Error al crear el fichero auxiliar.',13,10,0
msgerr2 DB `El directorio está vacío.',13,10,0
msgerr3 DB `Error de lectura en fichero.',13,10,0
msgerr4 DB `Error unidad lógica llena.',13,10,0
```

El código correspondiente será (no se pidió en examen):

```
error1: DISP_STRINGZ msgerr1
        END_PROCESS 1
error2: DISP_STRINGZ msgerr2
        CLOSE_HANDLE hdl_aux
        END_PROCESS 2
error3: DISP_STRINGZ msgerr3
        CLOSE_HANDLE hdl_aux
        CLOSE_HANDLE hdl_fich
        END_PROCESS 3
error4: DISP_STRINGZ msgerr4
        CLOSE_HANDLE hdl_aux
        CLOSE_HANDLE hdl_fich
        END_PROCESS 4
```

8) El programa pretende buscar una cadena de, como máximo, 16 caracteres en todos los ficheros del directorio actual. Contando e identificando los ficheros que contienen la cadena y presentando la posibilidad de imprimir el listado de ellos en pantalla.



Problema 2 final/1 parcial. (Junio tarde).

(3/5 Puntos).

El S.O. debe cargar un programa en memoria que está compuesto de los siguientes segmentos:

- Un segmento de código que ocupa 245B0H Bytes. La primera instrucción a ejecutar está en el offset 0.
- Un segmento de pila que ocupa 128 KBytes.
- Un segmento de datos (variables del programa) que ocupa 6225H Bytes.

Dado que es una aplicación de usuario, tendrá el menor privilegio posible (el que tenga menor posibilidad de acceder a los recursos del sistema).

Se deben alojar los tres segmentos, en el orden en el que se han mencionado, a partir de la dirección de memoria 20000H, utilizando posiciones consecutivas sin ningún hueco entre segmentos.

Se sabe también que el programa llama a una subrutina del S.O. que no recibe datos por pila y que comienza en el offset 150H de un segmento no conforme con DPL = 0, al que se accede con el selector 4564H.

Se pide:

- A) Crear un descriptor de CALL GATE para llamar a la subrutina del S.O. Se debe justificar el contenido de cada campo, y si alguno no pudiera calcularse, se pondrá a 0, indicándose expresamente. (0,5/0,8 pts.)
- B) Crear los descriptores de los tres segmentos del programa³. Se aplican los mismos requerimientos que en el apartado A). (1,0/2,0 pts.)
- C) El S.O. deberá crear una LDT para el programa que se está cargando. Dicha tabla se colocará en la dirección de memoria 3280H y tendrá el tamaño estrictamente necesario para la GATE y los tres descriptores de segmento, que, además, deberán ir colocados en este mismo orden. Crear el descriptor correspondiente a esta LDT, e indicar las direcciones en las que se aloja cada uno de los 4 descriptores que componen esta tabla. (0,7/1,0 pts.)
- D) Si la pila está vacía ¿Cuáles serán los valores iniciales de SS y (E)SP? (Se debe usar el puntero de pila que corresponda). (0,4/0,6 pts.)
- E) Valores de CS y (E)IP para comenzar la ejecución del programa si la primera instrucción a ejecutar está en el offset 0. Se debe usar el puntero de instrucción que corresponda. (0,4/0,6 pts.)

SOLUCIÓN

NOTA: Debe contestarse a cada apartado en el lugar destinado para él a partir de la próxima hoja.

³ En el examen final sólo se pedían los descriptores de código y pila, no el de datos.



Problema 2 final/1 parcial. (Junio tarde).

(Cont).

APARTADO A)

Campo	Valor	Justificación
Offset	150H	Dato del enunciado
P	1	Para evitar excepciones al acceder a la GATE
DPL	3	Para que se cumpla la regla $GATE\ DPL \geq EPL$, ya que $EPL=3$
2/3	0	Valor indefinido: no sabemos si la rutina llamada es de 16 o 32 bits, ni hay parámetros cuyo tipo (Word o Doble Word) nos lo indique
Tipo	C.GATE	Dato del enunciado (Crear un descriptor de CALL GATE ...)
Count	0	No se pasan parámetros por pila
Selector	4564H a 4567H	Dato del enunciado. Como el RPL no se usa hay 4 valores válidos para el selector.

Descriptor:	00 00 E4 00 45 64 01 50H
--------------------	--------------------------

APARTADO B)

Segmento de código:

Campo	Valor	Justificación
Dir.Base	20000H	Dato del enunciado
Gr.	0	No es posible ponerlo a 1 porque Límite no acaba en FFFH
Def	1	Tarea de 32 bits ya que tiene segmentos mayores de 64KB
Av.	0	No hay datos de este campo.
Límite	245AFH	Tamaño del segmento menos uno.
P	1	Se está cargando el programa en memoria.
DPL	3	Mínimo nivel de privilegio según enunciado.
Tipo	Seg.Cód	Dato del enunciado
C	0	No tiene sentido declararlo como conforme si el $DPL=3$
R	0	Campo no definido por el enunciado
A	0	Campo no definido por el enunciado

Descriptor:	00 42 F8 02 00 00 45 AFH
--------------------	--------------------------



Problema 2 final/1 parcial. (Junio tarde).

(Cont).

Segmento de pila:

Campo	Valor	Justificación
Dir.Base	445B0H	Dato del enunciado
Gr.	0	Puede valer 0 y 1. Según enunciado seleccionamos 0
Big	1	Pila de 32 bits ya que es mayor de 64KB
Av.	0	No hay datos de este campo.
Límite	1FFFFH	Tamaño del segmento menos uno.
P	1	Se está cargando el programa en memoria.
DPL	3	Debe ser igual al nivel de privilegio de la tarea.
Tipo	Seg.Datos	La pila debe estar en un segmento de datos (Tabla 7 Anexo E)
ED	0	No definido en enunciado. Se toma 0
W	1	El segmento de pila debe ser escribible. (Tabla 7 Anexo E)
A	0	Campo no definido por el enunciado

Descriptor:	00 41 F2 04 45 B0 FF FFH
--------------------	--------------------------

Segmento de datos:

Campo	Valor	Justificación
Dir.Base	645B0H	Dato del enunciado
Gr.	0	No es posible ponerlo a 1 porque Límite no acaba en FFFH
Big	0	Este bit no tiene significado en un segmento que no sea para la pila
Av.	0	No hay datos de este campo.
Límite	6224H	Tamaño del segmento menos uno.
P	1	Se está cargando el programa en memoria.
DPL	3	Debe ser igual o peor que el nivel de privilegio de la tarea.
Tipo	Seg.Datos	Es la única posibilidad de tener un segmento "escribible"
ED	0	En un segmento de datos no tiene sentido que sea 1.
W	1	Las variables de un programa deben ser modificables
A	0	Campo no definido por el enunciado



Problema 2 final/1 parcial. (Junio tarde).

(Cont).

Descriptor:	<i>00 00 F2 06 45 B0 62 24H</i>
--------------------	---------------------------------

APARTADO C)

Descriptor LDT:

Campo	Valor	Justificación
<i>Dir.Base</i>	<i>3280H</i>	<i>Dato del enunciado</i>
<i>Gr.</i>	<i>0</i>	<i>No es posible ponerlo a 1 porque Límite no acaba en FFFH</i>
<i>Av.</i>	<i>0</i>	<i>No hay datos de este campo.</i>
<i>Límite</i>	<i>1FH</i>	<i>Tamaño del segmento (4 descriptores de 8 bytes) menos uno.</i>
<i>P</i>	<i>1</i>	<i>Se está cargando la LDT en memoria.</i>
<i>Tipo</i>	<i>LDT</i>	<i>Dato del enunciado</i>

Descriptor:	<i>00 00 82 00 32 80 00 1FH</i>
--------------------	---------------------------------

Contenido LDT

Dirección	Contenido
<i>3280H a 3287H</i>	<i>Descriptor de Gate (0000E40045640150H)</i>
<i>3288H a 328FH</i>	<i>Descriptor de Código (0042F802000045AFH)</i>
<i>3290H a 3297H</i>	<i>Descriptor de Pila (0041F20445B0FFFFH)</i>
<i>3298H a 329FH</i>	<i>Descriptor de Datos (0000F20645B06224H)</i>

Direcciones ocupadas por la LDT: Desde 3280H hasta 329FH

APARTADO D)

*SS contendrá el selector que apunta al descriptor de la pila. sus campos serán:
Índice: 2. TI: 1 (LDT). RPL: 3 Igual al CPL de la tarea.*

SS = 17H.

Puntero de pila: ESP (pila de 32 bits). Apuntará una posición más allá de la primera posición libre. Es decir Límite + 1 = 20000H.

ESP = 20000H



Problema 2 final/1 parcial. (Junio tarde).

(Cont).

APARTADO E)

CS contendrá el selector que apunta al descriptor del segmento de código. sus campos serán:

Índice: 1. TI: 1 (LDT). RPL: 3 Igual al CPL de la tarea.

CS = 0FH.

Puntero de instrucción: EIP (tarea de 32 bits). Apuntará al offset de la primera instrucción: 0

EIP = 0.

Depto. de Electrónica y Comunicaciones de la U.P.S.A.M.



Problema 3 final/2 parcial. (Junio tarde).

(2/3 Puntos).

En un sistema basado en un microprocesador IA-32 con la paginación activada, se realiza la instrucción **MOV ES:[1384H],EAX**, que realiza un acceso de **escritura a una DWORD** cuya dirección lineal es: 962E4384H.

Se conocen además los siguientes datos referidos a ese mismo momento:

- A) La traducción de la página lineal correspondiente a la física no está incluida en el TLB.
- B) Está activado el direccionamiento extendido.
- C) La dirección de comienzo de la primera tabla implicada en el acceso es: 048312C00H.
- D) Se dispone del siguiente volcado de memoria:

DIRECCIÓN	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
048312C00H	3F	36	92	1C	0B	00	00	00	86	52	65	0E	06	00	00	00
048312C10H	19	BE	9A	57	03	00	00	00	52	3A	08	76	0C	00	00	00
123456710H	3A	B2	00	00	0F	00	00	00	58	F7	9C	49	0D	00	00	00
123456720H	15	05	CE	79	05	00	00	00	19	CA	3A	76	04	00	00	00
3579AB580H	68	DB	00	00	00	00	00	00	35	6A	45	23	01	00	00	00
3579AB590H	24	02	C4	43	92	00	00	00	90	8F	3A	7A	1F	00	00	00

Se pide (responda en los recuadros destinados al efecto, usando el resto del espacio disponible para las operaciones necesarias para desarrollar el ejercicio):

- 1) Dirección y contenido de las entradas de las tablas relacionadas con el acceso. (0,4/0,6 ptos.)

TABLA	DIRECCIÓN	CONTENIDO
PDPT	048312C10H	00000003579ABE19H
PD	3579AB588H	0000000123456A35H
PT	123456720H	0000000579CE0515H

- 2) Dirección física a la que se pretende acceder ¿Se llega a realizar el acceso? Justifique la respuesta. (0,4/0,6 ptos.)

Dirección física: **579CE0384H**

¿Se llega a realizar el acceso?

Los bits U/S y R/W valen 1 y 0 respectivamente tanto en la tabla PD como en la PT, con lo que el acceso en escritura sólo estará permitido si la tarea tiene un CPL < 3 (nivel supervisor), generándose una interrupción si CPL = 3



Problema 3 final/2 parcial. (Junio tarde).

(Cont).

- 3) Suponga, independientemente de la respuesta del apartado anterior, que el acceso se realiza. ¿Qué valores se guardan en el TLB como consecuencia de este acceso?. (0,3/0,4 pts.)

VALOR	CONTENIDO
Nº de página lineal	962E4H
Nº de página física	579CE0H
U/Se (1)	1
R/We (1)	0
G (2)	1
D (3)	1

(1) Observando los bits de control de las entradas de PD y PT.

(2) id. de PT.

(3) id. de PT y tras el acceso.

- 4) En el supuesto del apartado 3). ¿Se realiza alguna modificación en las entradas de las tablas involucradas en el proceso?. En caso afirmativo, indicar el (los) nuevo(s) valor(es) de la(s) entrada(s) que varíe(n). (0,3/0,4 pts.)

Observando los bits de control de PT se comprueba que D y A tienen valor 0 y deberán quedar a 1 tras el acceso, con lo que la entrada resulta: 0000000579CE0575H

- 5) ¿Cuál es la dirección base indicada en el descriptor de ES?. Justificar brevemente la respuesta. (0,2/0,4 pts.)

Si la paginación está activada, Dirección base + Offset segmentación = Dirección lineal. Por lo tanto, tenemos: Dirección base = Dirección lineal - Offset segmentación = 962E4384 - 1384H = 962E300H.



Problema 3 final/2 parcial. (Junio tarde).

(Cont).

- 6) Si antes de que se borren estos datos del TLB se intenta acceder a la dirección lineal 962E48ACH, indicar, en función del tipo de acceso (lectura, escritura o fetch) y del CPL en curso, cual de estas opciones ocurriría:
- Realiza la traducción a través de los datos del TLB y no accede ni al PD ni a la PT.
 - Necesita realizar un acceso a la PT, pero no tiene que pasar por el PD.
 - Se produce una excepción, y por lo tanto no se llega a realizar el acceso.
 - El acceso requiere leer y/o escribir previamente las entradas del PD y de la PT.
- (0,4/0,6 pts.)

-CPL < 3 (modo supervisor): Independientemente del tipo de acceso, la opción es la a), pues no hay restricciones y ya se ha realizado acceso de escritura.

-CPL = 3 (modo usuario): Tiene restringido el acceso de escritura, luego se produciría lo que indica la opción c). Para el resto de modos de acceso vuelve a suceder la opción a).

Depto. de Electrónica y Comunicaciones de la U.S.



Problema 4 final⁴/3 parcial. (Junio tarde).

(1/2 Puntos).

En las siguientes preguntas de tipo test, sólo existe una respuesta correcta, excepto en la última, en la que únicamente hay una respuesta falsa y las demás son verdaderas. Cada pregunta tiene un valor de 0,2 pts. Cada respuesta incorrecta descuenta 0,05 pts. La pregunta que esté en blanco no sumará ni restará puntos.

Notas: • Se debe poner una X sobre la letra de la tabla "SOLUCIÓN" (que encontrará en la última hoja) correspondiente a la respuesta correcta.

• La tabla "CALIFICACIÓN" no deberá utilizarla. Es para su posterior calificación.

• No se tendrán en cuenta las respuestas que no se encuentren en dicha tabla "SOLUCIÓN".

1) En un sistema con varias cachés, que comparten la misma memoria principal, ...

- a) ... todas las cachés deben seguir el mismo criterio de escritura.
- b) ... todas las cachés tienen el mismo tamaño en bytes.
- c) ... todas las cachés usan el mismo tamaño de línea.
- d) ... todas las cachés deben tener el mismo número de planos o vías.

2) Si una caché usa el protocolo MESI, para identificar el estado en el que se encuentran las líneas que tiene cargadas, ...

- a) ... deberá actualizar la línea en memoria principal siempre que se modifique.
- b) ... los accesos de escritura del micro actualizan la línea en memoria principal sólo si se encuentran con fallo de caché.
- c) ... ningún acceso de escritura del procesador provoca escritura simultánea en memoria principal.
- d) ... si en un acceso de escritura del micro hay fallo de caché, no se carga la línea y se modifica exclusivamente en memoria principal.

3) En una caché semiasociativa de n vías o planos (n>1):

- a) El número de vías depende del número de posiciones que tenga cada vía.
- b) El número de vías depende del tamaño de la memoria principal a la que está asociada.
- c) Puede no existir memoria de algoritmo.
- d) El tamaño de la memoria de algoritmo depende del número de posiciones de cada vía.

⁴ En el examen final había sólo cinco de estas diez preguntas (se eliminaron las cuestiones 2, 4, 6, 8 y 10).



Problema 4 final/3 parcial. (Junio tarde).

(Cont).

- 4) **Los parámetros de configuración de una caché (i, j y k):**
- Dependen del tamaño de memoria principal disponible en el equipo.
 - Dependen del tamaño de memoria principal conectable al equipo.
 - Son independientes entre sí para cualquier caché.
 - Son aleatorios para cualquier caché.
- 5) **En cuanto a los criterios de escritura de una caché:**
- El inmediato (write through) actualiza la memoria principal en los accesos de escritura del procesador sólo si hay fallo de caché.
 - El obligado (write back) escribe en memoria principal únicamente cuando el procesador lo solicita.
 - El diferido (posted-write through) actualiza la línea en todas las cachés que la compartan y, a continuación, actualiza también la memoria principal.
 - El obligado (write back) sólo escribe en memoria principal si detecta que otra caché pretende cargar una línea no actualizada o si descarga una línea modificada.
- 6) **En una memoria caché de 512KB, asociada a una memoria principal de 2GB, con una memoria de algoritmo de 4K posiciones y que usa una línea de 16B:**
- No hay datos suficientes para conocer el valor de los parámetros i, j y k.
 - $i=4, j=12$ y $k=15$
 - $i=4, j=15$ y $k=12$
 - $i=4, j=12$ y $k=16$
- 7) **En una caché directamente mapeada ...**
- ... cada línea de memoria principal tiene una posición de caché única, previamente asignada.
 - ... puede existir memoria de algoritmo.
 - ... el número de posiciones de la memoria de etiquetas puede ser diferente al número de líneas de la memoria de contenido.
 - ... dependiendo de la asignación realizada, una línea de memoria puede tener una posición única reservada en la caché o más de una.



Problema 1 final. (Septiembre mañana).

(4 Puntos).

Dado el programa en ensamblador de la página siguiente, se pide:

1. Diseñe la macro *mac1*, que cuenta con dos parámetros (*par1* y *par2*) para que cumpla las siguientes condiciones: (0,6 ptos)
 - a. Sacar por pantalla un mensaje ASCIIZ, que será el primer parámetro (*par1*).
 - b. Leer de pantalla una secuencia de caracteres cuyo tamaño, incluyendo el retorno de carro, es el segundo parámetro de la macro. La cadena recibida debe quedar almacenada en *var3*.
 - c. Realizar un cambio de línea en pantalla, tras recibir la cadena anterior.
 - d. Convertir la cadena recibida en ASCIIZ.
2. Teniendo en cuenta las características de la macro *mac1*: (0,8 ptos)
 - a. ¿Qué función realiza el fragmento de código identificado con el comentario *a*?
 - b. ¿Qué tipo de información se solicita en el mensaje *msg1*?
 - c. ¿En qué condiciones se presenta en pantalla en mensaje *msgerr0* y cuál es su significado para el programa? Realice la declaración más explícita posible de la variable *msgerr0*.
 - d. Al final del fragmento *a* ¿Qué quedará almacenado en *var8*?
3. Respecto al fragmento identificado con el comentario *b*. (0,6 ptos)
 - a. Realice la declaración más explícita posible de la variable *msg2*.
 - b. ¿Cuál es la finalidad de este fragmento dentro del programa? ¿Para qué se usa en este caso la macro MOVE_PTR?
 - c. ¿Qué tiene que ocurrir para que se produzca el salto a *eti4*? Responda en función de parámetros del programa.
4. ¿A qué posición se desplaza el puntero del fichero abierto en el fragmento *b*, tras la ejecución del fragmento *c*? (0,4 ptos)
5. Explique, en el contexto del programa, cuál es la función del bucle *bucle1*, identificado con el comentario *d*, especificando la condición o condiciones de permanencia en el bucle. (0,4 ptos)
6. Repita el apartado anterior con el *bucle2*, identificado por el comentario *e*. (0,4 ptos)
7. Conteste brevemente a las siguientes cuestiones: (0,8 ptos)
 - a. Respecto al fragmento identificado con el comentario *f*, ¿cuál es su finalidad dentro del programa?
 - b. Describa brevemente qué función realiza el programa en su conjunto.
 - c. Teniendo en cuenta los apartados anteriores, realice la declaración más explícita posible de las variables *msg1*, *msg3* y *msg4*.

NOTAS ACLARATORIAS

a) La declaración de las variables mensaje solicitadas en el enunciado debe ser lo más completa posible, incluyendo el salto a la línea siguiente.

b) La llamada a macro **RENAME_FILE nomfich1, nomfich2**, permite cambiar el nombre a un fichero cerrado, siendo **nomfich1** el nombre antiguo y **nomfich2** el nuevo (representados como cadenas ASCIIZ). A efectos del programa, considérese que esta macro está correctamente definida en **macros.mac**.

c) Ténganse en cuenta las siguientes equivalencias ASCII (en decimal):

Retorno de carro:	13
Fin de línea:	10
Null:	0



Problema 1 final. (Septiembre mañana).

(Cont).

INCLUDE macros.mac

Declaración de Mac1

```

MODEL      SMALL
STACK      200H
DATASEG
msg1       ----
msg2       ----
msg3       ----
msg4       ----
msgerr0    ----
msgerr1    DB      'Error al abrir fichero.',13,10,0
msgerr2    DB      'Error al crear fich.aux.',13,10,0
msgerr3    DB      'Error de lectura o escritura en fichero.',13,10,0
msgerr4    DB      'Error: Unidad lógica llena.',13,10,0
msgerr5    DB      'Error al borrar fichero.',13,10,0
cr_lf      DB      13,10,0
var1       DB      ?
var2       DB      ?
var3       DB      13 DUP (?)
var4       DB      "fich.aux",0
var5       DW      ?
var6       DW      ?
var7       DB      100H DUP (?)
var8       DD      ?
var9       DB      ?

CODESEG
P386N
EXTRN      atud:NEAR
           STARTUPCODE
eti1:      mac1      msg1,11      ;a
           PUSH      OFFSET var3   ;a
           PUSH      WORD PTR 10    ;a
           CALL      atud           ;a
           TEST      AX,AX          ;a
           JNS      eti2           ;a
           DISP_STRINGZ msgerr0     ;a
           JMP      eti1           ;a
eti2:      MOV      var8,ESI        ;a
           mac1      msg2,13        ;b
           OPEN_HANDLE var3,0       ;b
           JC      err1            ;b
           MOV      var5,AX         ;b
           MOVE_PTR  var5,0,0,2     ;b
           SHL      EDX,16          ;b
           MOV      DX,AX           ;b
           CMP      EDX,var8        ;b
           JBE      eti4           ;b
           MOV      EAX,var8        ;c
           INC      EAX             ;c
           NEG      EAX             ;c
           MOV      DI,AX           ;c
           SHR      AX,16           ;c
           MOV      SI,AX           ;c
           MOVE_PTR  var5,SI,DI,2   ;c
           CREATE_HANDLE var4,0     ;c
           JC      err2            ;c
           MOV      var6,AX         ;c
           bucle1:  READ_HANDLE var5,var9,1 ;d
                   JC      err3            ;d
                   CMP      AX,1         ;d
                   JNE      eti3            ;d
                   CMP      var9,13       ;d
                   JNE      bucle1         ;d
           bucle2:  READ_HANDLE var5,var7,100H ;e
                   JC      err3            ;e
                   OR      AX,AX         ;e
                   JZ      eti3          ;e
                   WRITE_HANDLE var6,var7,AX ;e
                   JC      err3            ;e
                   CMP      AX,CX        ;e
                   JNE      err4          ;e
           JMP      bucle2         ;e
           eti3:    CLOSE_HANDLE var5     ;f
                   CLOSE_HANDLE var6     ;f
                   DELETE_ENTRY var3     ;f
                   JC      err5          ;f
                   RENAME_FILE var4,var3 ;f
                   DISP_STRINGZ msg3     ;f
                   END_PROCESS 0         ;f
           eti4:    DISP_STRINGZ msg4     ;f
                   CLOSE_HANDLE var5     ;f
                   END_PROCESS 0         ;f
           err1:    DISP_STRINGZ msgerr1  ;f
                   END_PROCESS 1         ;f
           err2:    DISP_STRINGZ msgerr2  ;f
                   CLOSE_HANDLE var5     ;f
                   END_PROCESS 2         ;f
           err3:    DISP_STRINGZ msgerr3  ;f
                   JMP      eti5         ;f
           err4:    DISP_STRINGZ msgerr4  ;f
           eti5:    CLOSE_HANDLE var5     ;f
                   CLOSE_HANDLE var6     ;f
                   DELETE_ENTRY var4     ;f
                   END_PROCESS 3         ;f
           err5:    DISP_STRINGZ msgerr5  ;f
                   DELETE_ENTRY var4     ;f
                   END_PROCESS 4         ;f
                   END

```

**Problema 1 final. (Septiembre mañana).****(Cont).****SOLUCIÓN**

1. La macro solicitada podría ser:

```
mac1 MACRO      par1,par2
      DISP_STRINGZ par1
      GET_STRING  par2,var1
      DISP_STRINGZ cr_lf
      XOR        BX,BX
      MOV        BL,var2
      MOV        var3[BX],0
      ENDM
```

2. Recibe por teclado una cadena ASCII que corresponde a un valor decimal (solicitado por el mensaje `msg1`), y obtiene, mediante la rutina `atud`, el valor numérico correspondiente, almacenándolo en `var8`. El mensaje `msgerr0` se producirá si aparece algún error durante la ejecución de la rutina `atud` (carácter no numérico o desbordamiento), identificado por un valor negativo en `AX`.

```
msgerr0 DB 'Parámetro no válido.',13,10,0
```

3. En este fragmento se solicita por teclado el nombre de un fichero, que se abre a continuación. Mediante la macro `MOVE_PTR` se obtiene el tamaño del fichero abierto, almacenándolo en `EDX`. Posteriormente, se compara el tamaño del fichero con el valor recibido por teclado en el fragmento a. Si el tamaño es menor o igual que el valor, se salta a `eti4`, y si no, continúa la ejecución con el fragmento c. Una declaración posible de `msg2` sería:

```
msg2 DB 'Teclee el nombre del fichero.',13,10,0
```

4. Si llamamos `N` al valor proporcionado por teclado al comienzo del programa, el puntero se situará en `EOF-(N+1)`. Este fragmento finaliza creando el fichero "fich.aux", que posteriormente se usará como fichero auxiliar.
5. Desde la posición actual del puntero del fichero se van leyendo caracteres hasta encontrar un valor 13 (retorno de carro), y entonces se ejecuta el bucle 2. Se sale del bucle1 por 3 causas: si se encuentra retorno de carro, se continúa el programa (etiqueta bucle2); si se produce error de lectura de fichero, se salta a `err3` para tratar el error y salir, y si se llega a EOF sin encontrar el retorno de carro, se salta a `eti3` para salir sin error.
6. Se leen los caracteres que queden en el fichero abierto desde la posición siguiente al retorno de carro encontrado en bucle1 y se llevan al fichero auxiliar "fich.aux". La salida del bucle se produce por las siguientes causas: si se produce algún error al leer o escribir en los ficheros, se salta a `err3` para tratar el error y salir; si se produce error de disco lleno al escribir, se salta a `err4`, donde se tratará el error y se saldrá del programa, y si se llega a EOF, se continuará en `eti3` para salir sin error.



Problema 1 final. (Septiembre mañana).

(Cont).

7. Si se ejecuta el fragmento identificado con *eti3*, se cierran los dos ficheros, a continuación se borra el fichero inicial y se asigna su nombre al fichero auxiliar, sacando por pantalla un mensaje alusivo y finalizando.

Si se ejecuta la parte identificada por *eti4*, simplemente se cierra el fichero original, quedándose como estaba, se visualiza por pantalla un mensaje alusivo y se finaliza.

El programa comprueba si el tamaño de un fichero es mayor que un valor, tanto el nombre del fichero como el valor los proporciona el usuario. Si lo es, el programa trunca el fichero eliminando su parte inicial hasta el primer retorno de carro que permita que el tamaño restante se ajuste al valor indicado. Si no encuentra ningún retorno de carro, el fichero queda vacío.

Si el tamaño del fichero es menor o igual que el valor, no se modifica.

Una posible declaración de las variables *msg1*, *msg3* y *msg4* sería:

```
msg1 DB 'Teclee el tamaño máximo en Bytes'  
      DB '(Máximo 10 dígitos decimales).',13,10,0  
msg3 DB 'Fichero truncado correctamente.',13,10,0  
msg4 DB 'Fichero de tamaño correcto.',13,10,0
```

NOTA: Esta aplicación está pensada para controlar el crecimiento de ficheros de tipo "logs" (almacenamiento de eventos) para que no superen nunca un tamaño dado, pero de forma que los registros que se vayan eliminando sean los más antiguos (los primeros del fichero); se ha supuesto también que los registros son ASCII terminados en retorno de carro y de tamaño variable y, por eso, al truncar se busca el primer retorno de carro, para que no quede ningún registro incompleto. (No se pedía la explicación indicada en esta nota).



Problema 2 final/1 parcial⁵. (Septiembre mañana). (3/5 Puntos).

A) De un sistema basado en un microprocesador IA-32, con la paginación desactivada, se conocen los siguientes datos referidos a un determinado momento:

- Los contenidos de los registros CS y TR son los siguientes:

CS=

2A37H	¿?
-------	----

; TR=

¿?	01 00 89 00 00 00 00 A3H
----	--------------------------

- La GDT y la LDT comienzan, respectivamente, en las direcciones 00080000H y 01001000H.
- Los contenidos de las direcciones de memoria del siguiente volcado, en el que todos los valores están en hexadecimal:

Direcciones	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00FFFFFF0	00	00	00	00	50	13	01	00	68	24	00	00	03	69	CF	00
01000000	00	00	00	00	34	23	12	01	48	12	00	00	12	34	56	78
.....
01000060	10	00	00	00	00	00	A0	00	02	04	06	08	0A	0C	0E	01
01000070	12	23	34	45	56	67	78	89	9A	AB	BC	CD	DE	EF	F0	00
01000080	13	57	9B	DF	02	46	8A	CE	00	03	36	69	9C	CF	F2	25
01000090	58	8B	BE	E1	14	47	7A	AD	D0	00	00	00	00	00	00	00
010000A0	00	00	00	60	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Se pide:

- A1. CPL de la tarea. (0,2/0,3 pts.).
- A2. Direcciones iniciales y finales de las siguientes zonas del TSS: zona obligatoria, zona opcional para el S.O. y zona de I/O Map. (0,5/0,7 pts.).
- A3. Sabiendo que el IOPL es igual a 2, ¿a qué puertos podría acceder la tarea con el CPL actual?. (0,3/0,5 pts.).

B) Si en ese momento se ejecuta la instrucción CALL 1844H:0000, que llama, a través de una gate, a una rutina del S.O. de 32 bits que necesita que le pasen por pila 2 dobles words, y se sabe que, justo al terminar de ejecutar esta instrucción, CS y EIP contienen los siguientes valores:

CS=

3748H	02 D0 99 46 8A CE 00 FFH
-------	--------------------------

; EIP=

00012345H

Se pide:

- B1. Decodificar la parte invisible de CS. ¿Qué tamaño tiene el segmento apuntado por este descriptor?. ¿Qué CPL obtiene la tarea mientras se ejecuta la rutina?. (0,6/0,9 pts.).
- B2. Diseñar el descriptor de gate necesario para realizar la llamada. (0,6/0,9 pts.).
- B3. ¿Qué direcciones de memoria debería ocupar el descriptor calculado en el apartado anterior?. (0,0/0,5 pts.).
- B4. ¿Qué contenidos tendrán los registros SS (sólo la parte visible) y ESP justo al terminar de ejecutar la instrucción CALL 1844H:0000?. (0,6/0,9 pts.).
- B5. ¿A qué puertos podría acceder la tarea mientras se ejecuta la rutina?. (0,2/0,3 pts.).

NOTA: Debe contestarse a cada apartado en el espacio reservado para él, y es imprescindible justificar brevemente las respuestas cuando así se indique.

⁵ El apartado B3 se omitió en el examen final.



Problema 2 final/1 parcial. (Septiembre mañana).

(Cont).

SOLUCIÓN

Apartado A1

<i>CPL</i>	<i>Justificación</i>
3	El RPL de CS es 3 (se saca decodificando su selector —2A37H—), y el RPL de CS es siempre igual al CPL.

Apartado A2

<i>Zona</i>	<i>Dir. inicial</i>	<i>Dir. final</i>	<i>Justificación</i>
<i>Obligatoria</i>	<i>01000000H</i>	<i>01000067H</i>	<i>De la parte invisible de TR se deduce que apunta a una TSS de 32 bits (y por tanto, el tamaño de su zona obligatoria es de 68 bytes —offsets 0 a 67H—) y que su dirección base, que es la de comienzo, es la 01000000H.</i>
<i>Para el S.O.</i>	<i>01000068H</i>	<i>0100009FH</i>	<i>Sus direcciones inicial y final vienen determinadas por las otras zonas: comienza a continuación del final de la zona obligatoria y termina justo antes de la zona del I/O Map</i>
<i>I/O Map</i>	<i>010000A0H</i>	<i>010000A3H</i>	<i>En los offsets 66H y 67H del TSS (direcciones 01000066H y 01000067H) está el Base I/O, que es 00A0H, y por tanto, el I/O Map comienza en el offset 00A0H del TSS, que es la dirección 010000A0H. La dirección final viene dada por el offset límite de la parte invisible de TR, que es 000A3H, y se corresponde con la dirección 010000A3H.</i>



Problema 2 final/1 parcial. (Septiembre mañana).

(Cont).

Apartado A3

Puertos	Justificación
001DH y 001EH	Como $CPL < IOPL$, sólo puede acceder a los que tienen los bits correspondientes del I/O Map activos, y, como los únicos bits activos son el 5 y el 6 del byte 3, los puertos a los que puede acceder serán: $3 \cdot 8 + 5 = 1DH$ y $3 \cdot 8 + 6 = 1EH$.

Apartado B1

Campo	Valor
Dirección base	02468ACEH
Gr (granularidad)	1
Def	1 (segmento de 32 bits direccionado con EIP)
Av (para uso del S.O.)	1
Límite	000FFH (como $Gr=1$, Offset límite=000FFFFFFH)
P	1 (presente en memoria)
CPL	0
Tipo	Descriptor de segmento de código
C	0 (no conforme)
R	0 (no legible)
A	1 (recientemente accedido)

Tamaño del segmento: ; CPL:

Apartado B2

Campo	Valor	Justificación
Offset	00012345H	Offset al que se salta, que es el que queda en EIP.
P	1	Si no, habría excepción.
Gate DPL	3 (11 bin)	Para que cumpla que $Gate\ DPL \geq \text{Max}(CPL, RPL) = \text{Max}(3, 0)$.
Tipo	01100	Call gate de 32 bits.
D.C.	2	Lo dice el enunciado.
Selector	3748H, 3749H, 374AH o 374BH	Apunta al descriptor del segmento de código, y es el que queda cargado en la parte visible de CS, pero en el descriptor podría tener cualquier RPL, porque no se usa.

Descriptor:

(*) También podría ser 0001EC0237492345H, 0001EC02374A2345H o 0001EC02374B2345H



Problema 2 final/1 parcial. (Septiembre mañana).

(Cont).

Apartado B3

<i>Direcciones</i>	<i>Justificación</i>
01002840H a 01002847H	El selector que apunta a dicho descriptor es 1844H (el de la instrucción), que ocupará los offsets 1840H a 1847H de la LDT, y esta tabla, a su vez, comienza en la dirección 01001000H

Apartado B4

	<i>Justificación</i>
SS (p. visible) = <input type="text" value="2468H"/>	Como la call gate modifica el CPL, también cambia de pila, y el puntero inicial de la nueva pila para CPL=0 está en los offsets 4 a 0BH del TSS, y, por tanto, es 2468H:00011350H y, como el proceso escribe 6 dobles words (18H bytes) en la nueva pila (los antiguos SS y ESP; los dos parámetros; CS, y EIP), el valor final de ESP será: 00011350H - 18H = 00011338H.
ESP = <input type="text" value="00011338H"/>	

Apartado B5

Como CPL>IOPL, está permitido el acceso a todos los puertos (desde el 0000 hasta el FFFFH).



Problema 3 final/2 parcial. (Septiembre mañana). (2/3 Puntos).

En un sistema basado en un microprocesador IA-32 con la paginación activada, se realiza un acceso de **lectura de una DWORD** cuya dirección lineal es: 12345678H.

Se conocen además los siguientes datos referidos a ese mismo momento:

- A) La traducción de la página lineal correspondiente a la física no está incluida en el TLB.
- B) No está activado el direccionamiento extendido.
- C) El contenido de CR3 es 5AC43018H.
- D) Se dispone del siguiente volcado de memoria:

DIRECCIÓN	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
5AC43120H	1F	5A	65	0E	0B	00	00	00	BF	06	80	1C	06	00	00	00
5AC43130H	27	B4	A9	62	08	00	00	00	52	3A	08	76	0C	00	00	00
8AF35660H	3A	B2	00	00	0F	00	00	00	58	F7	9C	49	0D	00	00	00
8AF35670H	52	9B	00	00	0C	CA	3A	76	78	56	34	12	04	00	00	00
0E655D10H	68	DB	00	00	59	5B	F3	8A	00	00	FF	B7	A2	87	4C	D7
0E655D20H	37	21	00	00	04	00	00	00	00	EF	CD	AB	00	00	00	00

Se pide:

- 1) Dirección inicial de la tabla, dirección inicial de la entrada a la que se accede y contenido de dicha entrada, para todas las tablas relacionadas con el acceso. (0,5/0,75 ptos).
- 2) Dirección física a la que se realiza el acceso y valor leído. A partir de estos datos indica el valor o valores posibles para el bit PSE y para el CPL. (0,5/0,75 ptos).
- 3) ¿Qué valores se guardan en el TLB como consecuencia de este acceso?. (0,5/0,75 ptos).
- 4) En el supuesto del apartado 3). ¿Se realiza alguna modificación en las entradas de las tablas involucradas en el proceso?. En caso afirmativo, indicar el (los) nuevo(s) valor(es) de la(s) entrada(s) que varíe(n). (0,5/0,75 ptos).

NOTA: Solo se evaluarán las respuestas que aparezcan en las tablas de respuestas de la hoja 3 de este ejercicio.

SOLUCIÓN

Apartado 1)

Tabla	Dir. Inicial	Dir. Accedida	Contenido
PD	5AC43000H	5AC43120H	0E655A1FH
PT	0E655000H	0E655D14H	8AF35B59H

Apartado 2)

Dir. acceso de lectura	Contenido
8AF35678H	12345678H

PSE: X. No puede deducirse de los datos del enunciado.
CPL: 0, 1 ó 2. No está permitido el acceso en modo usuario.



Problema 3 final/2 parcial. (Septiembre mañana).

(Cont).

Apartado 3)

<i>Item en TLB</i>	<i>Valor</i>
<i>Dir. lineal</i>	12345H
<i>Dir. física</i>	8AF35H
<i>G</i>	1
<i>D</i>	1
<i>U/S</i>	0
<i>R/W</i>	0

Apartado 4)

<i>Tabla</i>	<i>Nuevo contenido</i>
<i>PD</i>	0E655A3FH
<i>PT</i>	8AF35B79H

Depto. de Electrónica y Comunicaciones de la U.P.S.A.M.



Problema 4 final⁶/3 parcial. (Septiembre mañana). (1/2 Puntos).

En las siguientes preguntas de tipo test, sólo existe una respuesta correcta, excepto en la última, en la que únicamente hay una respuesta falsa y las demás son verdaderas. Cada pregunta tiene un valor de 0,2 pts. Cada respuesta incorrecta descuenta 0,05 pts. La pregunta que esté en blanco no sumará ni restará puntos.

Notas: • Se debe poner una X sobre la letra de la tabla “SOLUCIÓN” (que encontrará en la última hoja) correspondiente a la respuesta correcta.

• La tabla “CALIFICACIÓN” no deberá utilizarla. Es para su posterior calificación.

• No se tendrán en cuenta las respuestas que no se encuentren en dicha tabla “SOLUCIÓN”.

1) En la caché semiasociativa de n vías puede afirmarse que...

- a) ...sólo existe un lugar posible para cada línea.
- b) ... existen n lugares posibles para almacenar una línea.
- c) ... las líneas se pueden almacenar en cualquier zona de la caché que se encuentre libre..
- d) ... ninguna de las anteriores es cierta.

2) Del protocolo MESI puede afirmarse que...

- a) ...es igualmente adecuado para políticas de escritura write back que write through.
- b) ... no es adecuado para la política de escritura write through.
- c) ... no es adecuado para la política de escritura write back.
- d) ... ninguna de las anteriores es cierta.

3) En un ordenador que permite una memoria principal de hasta 8 GB. con un tamaño de línea de 32 bytes y una caché con una memoria de algoritmo de 64 k posiciones podemos afirmar que ...

- a) ... la memoria de etiquetas tendrá 128 Kposiciones si la caché dispone de dos vías.
- b) ... la memoria de etiquetas tendrá posiciones de 13 + estado bits.
- c) ... la memoria de etiquetas tendrá posiciones de 14 + estado bits
- d) ... la memoria de etiquetas tendrá 128 Kposiciones si la caché dispone de cuatro vías.

4) En un sistema multiprocesador las cachés de todos los procesadores deben tener:

- a) El mismo criterio de escritura.
- b) El mismo tamaño de memoria de etiquetas.
- c) El mismo número de planos.
- d) La suma de los parámetros $i+j+k$ debe ser igual.

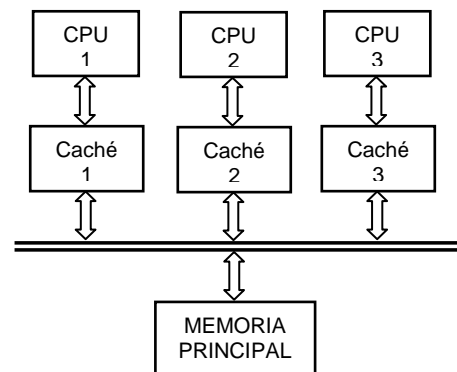
⁶ En el examen final había sólo cinco de estas diez preguntas (1, 3, 5, 8 y 10).



Problema 4 final/3 parcial. (Septiembre mañana). (Cont).

- 5) El principal inconveniente del criterio de escritura *write back* (escritura obligada) es el siguiente:
- a) Es más lento, ya que los ciclos de escritura del procesador deben hacerse a la velocidad de la memoria principal.
 - b) Satura más el bus.
 - c) Obliga a un *snooping* y a un protocolo de coherencia más complejos.
 - d) Ninguna de las anteriores es cierta.
- 6) En un ordenador que permite una memoria principal de hasta 1 GB tenemos un sistema compuesto por una caché de datos de 512KB y una memoria de etiquetas de 16K15 (16K posiciones de 15 bits). Si sabemos, además, que usa el protocolo de coherencia MESI y el algoritmo LRU, podemos afirmar que...
- a) ... la memoria de algoritmo es de 4K6.
 - b) ... la memoria de algoritmo es de 16K4.
 - c) ... la memoria de algoritmo es de 16K6.
 - d) ... ninguna de las anteriores es cierta..
- 7) Si en un determinado sistema de caché $i=5$, $j=12$, $k=14$ y $n=4$, la línea que comienza en la dirección 7A4C9820H puede estar cargada también en la siguiente dirección de la caché:
- a) 39820H.
 - b) 69820H.
 - c) 1820H.
 - d) Ninguna de las anteriores es posible.

Nota: Las siguientes preguntas de test se refieren a un sistema multiprocesador como el que aparece en la figura, del que se sabe que las *cachés 1 y 2* usan protocolo *MESI*, y *la 3* sigue el criterio de *escritura inmediato (write through)*, con un bit para especificar el estado de la línea.



- 8) Si la caché 3 (la que funciona con *write through*) contiene una línea, y está actualizada:
- a) Ninguna de las demás cachés puede contener esa misma línea simultáneamente.
 - b) Si alguna de las demás cachés contienen esa misma línea, no la tiene actualizada, es decir, la tiene en estado **I**.
 - c) Sólo una de las otras cachés puede contener esa línea.
 - d) Si alguna de las demás cachés contienen esa misma línea, puede que la tengan en estado **I** o en estado **S**.



Problema 1 final. (Septiembre tarde).

(4 Puntos).

Dado el siguiente programa:

INCLUDE macros.mac

Declaración de la macro Mac1

Declaración de la macro Mac2

```

MODEL          SMALL
STACK          200H
DATASEG
var1           DB      ?
var2           DB      ?
var3           DB      80 DUP(?)
var4           DW      ?
var5           DD      ?
var6           DD      1
var7           DB      10,13,0
msg1           DB      .....
msg2           DB      .....
msgerr1        DB      .....
msgerr2        DB      .....
msgerr3        DB      'Error al escribir fichero o disco lleno',10,13,0
CODESEG
P386N
EXTRN          atud:NEAR
STARTUPCODE
eti1:          mac1      msg1,11      ;A
               PUSH     OFFSET var3  ;A
               PUSH     WORD PTR 10   ;A
               CALL     atud          ;A
               CMP      AX,0          ;A
               JG       eti2          ;A
               DISP_STRINGZ msgerr1   ;A
               JMP      eti1          ;A
eti2:          MOV      var5,ESI      ;A
eti3:          mac1      msg2,80      ;B
               CREATE_HANDLE var3,0   ;B
               JNC      eti4          ;B
               DISP_STRINGZ msgerr2   ;B
               JMP      eti3          ;B
eti4:          MOV      var4,AX       ;B
eti5:          mac2
               MOV      EAX,var6     ;C
               CMP      EAX,var5     ;C
               JE       eti8         ;C
               CMP      EAX,3        ;C
               JE       eti6         ;C
               INC      var6         ;C
               JMP      eti5         ;C
eti6:          ADD      var6,2        ;D
               MOV      EAX,var6     ;D
               CMP      EAX,var5     ;D
               JA       eti8         ;D
               MOV      EBX,3        ;D
eti7:          MOV      EAX,var6     ;D
               XOR      EDX,EDX      ;D
               DIV      EBX          ;D
               OR       EDX,EDX      ;D
               JZ       eti6         ;D
               ADD      EBX,2        ;D
               CMP      EBX,EAX     ;D
               JNA      eti7         ;D
               mac2
               JMP      eti6         ;D
eti8:          CLOSE_HANDLE var4
               END_PROCESS 0
eti9:          DISP_STRINGZ msgerr3
               CLOSE_HANDLE var4
               END_PROCESS 1
END
    
```



Problema 1 final. (Septiembre tarde).

(Cont).

Se pide:

- 1). Codificar la macro **mac1**, que recibe como parámetros el nombre de variable de un mensaje y un número, para que realice las siguientes acciones:
 - Visualizar el mensaje que se ha recibido como primer parámetro.
 - Leer de teclado un *string* que tenga, como máximo, el número de caracteres indicado por el segundo parámetro, incluyendo en ese número el retorno de carro final. El *string* debe quedar en la variable **var3**.
 - Convertir el string recibido a ASCIIZ.
 - Realizar un salto de línea en la pantalla. (0,5 pts.).
- 2). Codificar la macro **mac2**, que no recibe ningún parámetro, para que escriba el contenido de **var6** en el fichero que se ha creado y, si se produce error de escritura o de disco lleno, salte a **eti9**. (0,4 pts.).
- 3). ¿Qué hace el fragmento que tiene como comentario **;A?**. Indique con precisión en qué casos se salta a **eti2** y en cuáles se vuelve a **eti1**. Codifique **msg1** y **msgerr1** con mensajes lo más concretos que sea posible. (0,6 pts.).
- 4). ¿Qué hace el fragmento que tiene como comentario **;B?**. Codifique **msg2** y **msgerr2** con mensajes precisos. (0,4 pts.).
- 5). ¿Qué hace el fragmento que tiene como comentario **;C?**. En función del valor que tenga inicialmente **var5**: ¿cuántas veces escribe en el fichero?; ¿qué valores escribe?; ¿en qué casos salta a **eti8**, y en cuáles continúa en **eti6**?. (0,5 pts.).
- 6). ¿Qué hace el fragmento que tiene como comentario **;D?**. (1,0 pts.).
- 7). ¿Qué hace el programa en conjunto?. (0,6 pts.).

NOTA: Debe contestarse cada apartado en el lugar destinado para él.

SOLUCIÓN

Apartado 1

```
mac1 MACRO mensaje,nrocaracteres
      DISP_STRINGZ mensaje
      GET_STRING nrocaracteres,var1
      DISP_STRINGZ var7
      MOV BL,var2
      XOR BH,BH
      MOV var3[BX],0
ENDM
```

Apartado 2

```
mac2 MACRO
      WRITE_HANDLE var4,var6,4
      JC eti9
      CMP AX,CX
      JNE eti9
ENDM
```



Problema 1 final. (Septiembre tarde).

(Cont).

Apartado 3

Fragmento “;A”

El fragmento pide por teclado una cadena, que representa un número entero sin signo en ASCII decimal, de 10 caracteres máximo (11 contando el retorno de carro final), la convierte a ASCIIZ y después a binario, dejando el resultado, que es una doble word, en `var5`. Se comprueba que el número obtenido no sea 0, ni sobrepase el tamaño de la doble word, ni que la cadena contenga algún carácter no numérico, y en cualquiera de estos tres casos, se visualiza `msgerr2` y se vuelve a `eti1` para pedir la cadena, y si no, se continúa a partir de `eti2`.

Codificación de `msg1` y `msgerr1`:

```
msg1      DB '¿Hasta qué número desea llegar?: ',0
msgerr1   DB 'Nº inválido (0, desbordamiento o car. inválido)',10,13,0
```

Apartado 4

Fragmento “;B”

El fragmento pide un nombre de fichero y lo intenta crear como fichero normal. Si lo logra, guarda el handle en `var4` y continúa a partir de `eti5`, y si no, visualiza `msgerr2` y salta a `eti3` para pedir de nuevo el nombre de fichero.

Codificación de `msg2` y `msgerr2`:

```
msg2      DB '¿En qué fichero quiere guardarlos?: ',0
msgerr2   DB 'Error al crear el fichero',10,13,0
```

Apartado 5

Escribe en el fichero números binarios consecutivos con tamaño de doble word, empezando por el 1. Termina cuando llega al valor de `var5` (el número pedido por el teclado) o a 3, lo que ocurra primero. Si lo primero que sucede es que llega a `var5` (lo cual quiere decir que `var5` es menor o igual a 3), después de escribir los números correspondientes, salta a `eti8` y termina el programa; en cambio, si llega a 3 (señal de que `var5` es mayor que 3), continúa ejecutando a partir de `eti6`. Por lo tanto, en este fragmento escribe:

Valor de <code>var5</code>	Escribe en fichero...	Después sigue en...
<code>var5=1</code>	1	<code>eti8</code> (termina)
<code>var5=2</code>	1,2	<code>eti8</code> (termina)
<code>var5=3</code>	1,2,3	<code>eti8</code> (termina)
<code>var5>3</code>	1,2,3	<code>eti6</code> (continúa)



Problema 1 final. (Septiembre tarde).

(Cont).

Apartado 6

El fragmento tiene dos bucles anidados (eti6:-JZ eti6 0 JMP eti6, que es el externo, y eti7:-JNA eti7, que es el interno).

El externo, lo único que hace es sumar 2 a var6 (recuérdese que al terminar el fragmento anterior tenía el valor 3) y a continuación comprobar si var6 ha sobrepasado a var5, y en ese caso salir a eti8 para terminar el programa; si no, inicializa EBX a 3 y entra en el bucle interno. Por tanto, cada vez que pase por el bucle externo, se entrará al interno con valores de var6 impares consecutivos menores de var5; por ejemplo, si var5=12, se entrará 4 veces con var6=5, 7, 9 y 11, y cuando var6 sea 13, se saldrá del bucle externo sin entrar en el interno. Cada vez que entra en el interno, EBX quedará inicializado a 3. El interno lo que hace es comprobar si var6 es divisible por 3; si lo es, sale, y si no, comprueba si es divisible por el siguiente impar (5), y así sucesivamente por 7, 9, 11..., hasta que el cociente de la división sea menor que el divisor, momento en el que podemos asegurar que var6 es primo. En ese caso, se escribe var6 en el fichero y se vuelve al bucle externo, mientras que si una división da exacta (var6 no es primo), se sale del bucle interno al externo sin escribir el número en el fichero.

Por lo tanto, y, en resumen, lo que hace este fragmento es buscar todos los números primos comprendidos entre 5 y var5 (ambos inclusive) y escribirlos en el fichero creado, en formato de dobles words binarias.

Apartado 7

El programa crea una tabla de números primos menores o iguales a un valor dado, empezando por el 1, y la escribe en formato de dobles words binarias, en un fichero que el propio programa crea. Pide por teclado el valor numérico límite hasta el que hay que llegar, y el pathname del fichero que tiene que crear. Maneja los siguientes errores:

- *Si el número límite que se introduce por teclado es incorrecto (es 0, no cabe en una doble word o tiene algún carácter no numérico), visualiza un error y vuelve a pedirlo.*
- *Si no puede crear el fichero tecleado, visualiza un error y vuelve a pedirlo.*
- *Si al escribir en el fichero se produce error de escritura o de disco lleno, se visualiza un error, se cierra el fichero y se termina con código de retorno 1.*



Problema 2 final/1 parcial. (Septiembre tarde).

(3/5 Puntos).

El S.O. debe cargar un programa en memoria que está compuesto de los siguientes segmentos:

- Un segmento de código que ocupa 24500H Bytes. La primera instrucción a ejecutar está en el offset 0.
- Un segmento de datos (variables del programa) que ocupa 6220H Bytes.
- Un segmento de datos (constantes del programa que no cambian nunca) que ocupa 80H bytes.

Dado que es una aplicación del S.O. tendrá el máximo privilegio posible (el que tenga mayor posibilidad de acceder a los recursos del sistema).

Se deben alojar los tres segmentos mencionados en el orden que aparecen, a partir de la dirección de memoria 40000H, utilizando posiciones consecutivas sin ningún hueco entre cada segmento.

Sus descriptores se almacenan en la GDT en la posición (índice) 123H y siguientes.

El programa contiene una subrutina con el offset de entrada 150H, que debe poder ser llamada desde cualquier otro programa cargado en el sistema. Dicha rutina debe recibir como parámetros 2 Words por pila.

Se pide:

- A) Crear los descriptores de los tres segmentos del programa.⁷ Si algún campo no puede calcularse se pondrá a 0 indicándose expresamente. (1,2/2,4 pts.).
- B) Crear un descriptor de CALL GATE para llamar a la subrutina del S.O. Se debe justificar el contenido de cada campo. Se aplican los mismos requerimientos que en el apartado A). (0,6/0,8 pts.).
- C) El S.O. ahora debe cargar un programa de usuario en memoria con 3 segmentos. Los descriptores de esos tres segmentos una vez cargados son: 00 42 F8 02 00 00 45 AFH, 00 41 F2 04 45 B0 FF FFH y 00 00 F2 06 45 B0 62 24H. Crea una tabla LDT con los siguientes requisitos:
- La tabla se colocará en la dirección de memoria 3280H.
 - Tendrá el tamaño estrictamente necesario para los tres descriptores de segmento y la GATE del apartado B)
 - Indica cuál será el contenido de las entradas de la LDT y sus direcciones.
- (0,3/0,4 pts.).
- D) Crear el descriptor de la LDT recién creada. Si algún campo no puede calcularse se pondrá a 0 indicándose expresamente. (0,6/0,8 pts.).
- E) Valores de CS y (E)IP al comenzar la ejecución de la subrutina para la que se diseñó la CALL GATE. (0,3/0,6 pts.).

NOTA: Debe contestarse cada apartado en el lugar destinado para él.

⁷ En el final sólo se pedían los descriptores de los segmentos de código y de datos variables. El de constantes no se pedía.



Problema 2 final/1 parcial. (Septiembre tarde).

(Cont).

SOLUCIÓN

APARTADO A)

Segmento de código:

<i>Campo</i>	<i>Valor</i>	<i>Justificación</i>
<i>Dir.Base</i>	<i>40000H</i>	<i>Dato del enunciado</i>
<i>Gr.</i>	<i>0</i>	<i>No es posible ponerlo a 1 porque Límite no acaba en FFFH</i>
<i>Def</i>	<i>1</i>	<i>Tarea de 32 bits ya que tiene segmentos mayores de 64KB</i>
<i>Av.</i>	<i>0</i>	<i>No hay datos de este campo.</i>
<i>Límite</i>	<i>244FFH</i>	<i>Tamaño del segmento menos uno.</i>
<i>P</i>	<i>1</i>	<i>Se está cargando el programa en memoria.</i>
<i>DPL</i>	<i>0</i>	<i>Máximo nivel de privilegio según enunciado.</i>
<i>Tipo</i>	<i>Seg.Cód</i>	<i>Dato del enunciado</i>
<i>C</i>	<i>0</i>	<i>Impondrá su nivel de Priv. para acceder a elementos del Sistema</i>
<i>R</i>	<i>0</i>	<i>Campo no definido por el enunciado</i>
<i>A</i>	<i>0</i>	<i>Campo no definido por el enunciado</i>

Descriptor:	<i>00 42 98 04 00 00 44 FFH</i>
--------------------	---------------------------------

Segmento de datos (variables)

<i>Campo</i>	<i>Valor</i>	<i>Justificación</i>
<i>Dir.Base</i>	<i>64500H</i>	<i>Dir. Base segm código + su tamaño. 40000H+24500H</i>
<i>Gr.</i>	<i>0</i>	<i>No es posible ponerlo a 1 porque Límite no acaba en FFFH</i>
<i>Big</i>	<i>0</i>	<i>Sólo es significativo para un segmento de pila</i>
<i>Av.</i>	<i>0</i>	<i>No hay datos de este campo.</i>
<i>Límite</i>	<i>621FH</i>	<i>Tamaño del segmento menos uno.</i>
<i>P</i>	<i>1</i>	<i>Se está cargando el programa en memoria.</i>
<i>DPL</i>	<i>0</i>	<i>Debe ser igual al nivel de privilegio de la tarea.</i>
<i>Tipo</i>	<i>Seg.Datos</i>	<i>Dato del enunciado</i>
<i>ED</i>	<i>0</i>	<i>Puede valer 1 ó 0. No definido en enunciado. Se toma 0</i>
<i>W</i>	<i>1</i>	<i>Segmento escribible dado que contiene variables del programa</i>
<i>A</i>	<i>0</i>	<i>Campo no definido por el enunciado</i>

Descriptor:	<i>00 40 92 06 45 00 62 1FH</i>
--------------------	---------------------------------



Problema 2 final/1 parcial. (Septiembre tarde).

(Cont).

Segmento de datos (Constantes)

<i>Campo</i>	<i>Valor</i>	<i>Justificación</i>
<i>Dir.Base</i>	6A720H	<i>Dir. Base = Dir Base segm. anterior+ su tamaño. 64500H + 6220H</i>
<i>Gr.</i>	0	<i>No es posible ponerlo a 1 porque Límite no acaba en FFFH</i>
<i>Big</i>	0	<i>Este bit no tiene significado en un segmento que no sea para la pila</i>
<i>Av.</i>	0	<i>No hay datos de este campo.</i>
<i>Límite</i>	7FH	<i>Tamaño del segmento menos uno.</i>
<i>P</i>	1	<i>Se está cargando el programa en memoria.</i>
<i>DPL</i>	0	<i>Debe ser igual al nivel de privilegio de la tarea.</i>
<i>Tipo</i>	Seg.Datos	<i>Dato del enunciado</i>
<i>ED</i>	0	<i>En un segmento de datos no tiene sentido que sea 1.</i>
<i>W</i>	0	<i>Las constantes de un programa no deben ser modificadas</i>
<i>A</i>	0	<i>Campo no definido por el enunciado</i>

Descriptor:	00 00 90 06 A7 20 00 7FH
--------------------	--------------------------

APARTADO B)

<i>Campo</i>	<i>Valor</i>	<i>Justificación</i>
<i>Offset</i>	150H	<i>Dato del enunciado</i>
<i>P</i>	1	<i>Para evitar excepciones al acceder a la GATE</i>
<i>DPL</i>	3	<i>Para que sea accesible con EPL=3 (regla aplicable GATE DPL >= EPL)</i>
<i>2/3</i>	0	<i>Los parámetros son de tipo WORD</i>
<i>Tipo</i>	C.GATE	<i>Dato del enunciado (Crear un descriptor de CALL GATE ...)</i>
<i>Count</i>	2	<i>2 parámetros por pila (Enunciado)</i>
<i>Selector</i>	918H	<i>Dato del enunciado. Como el RPL no se usa hay 4 valores válidos para el selector. Elegimos 918H por tener a 0 los bits menos signif.</i>

Descriptor:	00 00 E4 02 09 18 01 50H
--------------------	--------------------------



Problema 2 final/1 parcial. (Septiembre tarde).

(Cont).

APARTADO C)

Contenido LDT

Dirección	Contenido
3280H	00 42 F8 02 00 00 45 AFH (segmento 1)
3288H	00 41 F2 04 45 B0 FF FFH (segmento 2)
3290H	00 00 F2 06 45 B0 62 24H (segmento 3)
3298H	00 00 E4 02 09 18 01 50H (CALL GATE)

Direcciones ocupadas por la LDT: Desde 3280H hasta 329FH

APARTADO D)

Descriptor LDT:

Campo	Valor	Justificación
Dir.Base	3280H	Dato del enunciado
Gr.	0	No es posible ponerlo a 1 porque Límite no acaba en FFFH
Av.	0	No hay datos de este campo.
Límite	1FH	Tamaño del segmento (4 descriptores de 8 bytes) menos uno.
P	1	Se está cargando la LDT en memoria.
Tipo	LDT	Dato del enunciado

Descriptor:	00 00 82 00 32 80 00 1FH
-------------	--------------------------

APARTADO E)

CS contendrá en la parte visible el selector que tenemos en la CALL GATE = 918H. CS contendrá en la parte invisible el descriptor calculado en A) 00 42 98 04 00 00 44 FFH.

CS P. visible = 918H P. invisible = 00 42 98 04 00 00 44 FFH

Puntero de instrucción: EIP (tarea de 32 bits). Apuntará al offset de comienzo de la subrutina que es 150 H.

EIP = 150H



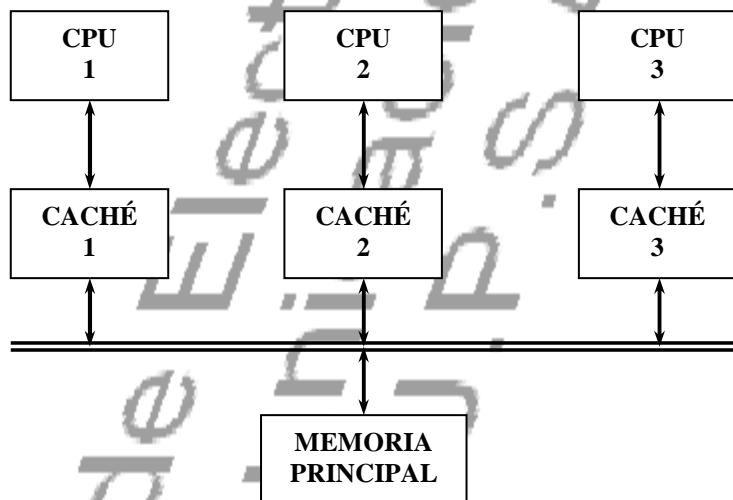
Problema 3 final/2 parcial. (Septiembre tarde).

(2/3 Puntos).

En un sistema multiprocesador como el de la figura, diseñado para un **tamaño máximo de memoria principal de 2GB**, se conocen los siguientes datos:

	CACHÉ 1	CACHÉ 2	CACHÉ 3
Capacidad de la caché de datos	128KB		
Memoria de etiquetas ^(*) (Nº de posiciones x nº de bits)	4K x 18	16K x 14	16K x 15
Criterio de Escritura y protocolo	W. back (MESI)	W. back (MESI)	W. back (MESI)

^(*)Incluyendo bits de estado



Se sabe también, que los **2 bits más significativos** de las memoria de etiquetas, representan el estado de la línea (protocolo MESI) según la tabla siguiente:

ESTADO	CÓDIGO
M	11
E	10
S	01
I	00

Se pide:

a) Calcular los valores de *i*, *j*, *k* y *n* para las 3 cachés.

(0,4/0,6 ptos)

	CACHÉ 1	CACHÉ 2	CACHÉ 3
<i>i</i>	5	5	5
<i>j</i>	10	14	13
<i>k</i>	16	12	13
<i>n</i>	4	1	2



Problema 3 final/2 parcial. (Septiembre tarde).

(Cont).

- b) Tamaño de la memoria de algoritmo (nº de posiciones y nº de bits por posición) de las caches 1, 2 y 3, sabiendo que usan el algoritmo LRU. (0,4/0,6 pts).

	Caché 1	Caché 2	Caché 3
Tamaño memoria de algoritmo (Nº de posiciones x nº de bits)	1K6	No tiene	8Kb

- c) Sabiendo que en un momento determinado la línea de memoria principal que comienza en la dirección 3A6C9EC0H está también cargada en el plano 0 de la caché 0 con estado E y en el último plano de la caché 3 con estado I, pero no está cargada en la caché 2:
- Indicar el rango de direcciones (inicial y final) de la línea correspondiente en las dos cachés en las que está cargada. (0,4/0,6 pts).
 - Especificar la dirección y el contenido de la memoria de etiquetas correspondientes a la línea cargada en las cachés 1 y 3. (0,4/0,6 pts).

		Memoria Principal	Caché 1	Caché 2	Caché 3
Dirección inicial		3A6C9EC0H	01EC0H		49EC0H
Dirección final			01EDFH		49EDFH
Nº de plano o vía			0		Último
Estado de la línea			E	No cargada	I
Memoria de etiquetas	Dirección		0F6H		24F6H
	Contenido		274D9H		0E9BH

- d) En estas condiciones, la CPU 2 realiza una operación de lectura en esa línea. Indicar qué acciones se realizan en la memoria principal, memoria de datos, y memoria de etiquetas por parte de cada una de las cachés. Si alguna etiqueta cambia su contenido debe indicarse el nuevo valor. (0,4/0,6 pts).

	Caché 1	Caché 2	Caché 3
Acciones en la memoria principal	---	Caché 2 pide la línea mediante BusRd y la memoria principal se la proporciona	---
Acciones en la caché de datos	---	Una vez recibida la línea, el procesador realiza la lectura	---
Acciones en la caché de etiquetas	S → Estado	S → Estado	No cambia. Sigue siendo I
Nuevo valor etiqueta	174D9H	174DH	0E9BH (no cambia)



Problema 4 final⁸/3 parcial. (Septiembre tarde).

(1/2 Puntos).

En las siguientes preguntas de tipo test, sólo existe una respuesta correcta. Cada pregunta tiene un valor de 0,2 ptos. Cada respuesta incorrecta descuenta 0,05 ptos. La pregunta que esté en blanco no sumará ni restará puntos.

Notas: • Se debe poner una X sobre la letra de la tabla “SOLUCIÓN” (que encontrará en la última hoja) correspondiente a la respuesta correcta.

• La tabla “CALIFICACIÓN” no deberá utilizarla. Es para su posterior calificación.

• No se tendrán en cuenta las respuestas que no se encuentren en dicha tabla “SOLUCIÓN”.

1) En los micros IA-32...

- a) ...el espacio de las direcciones lineales es de 64GB o de 4GB según esté o no activado el direccionamiento extendido.
- b) ...el espacio de direccionamiento lineal es único, y se comparte entre todas las tareas.
- c) ...en direccionamiento extendido, la tabla PDPT debe estar obligatoriamente en memoria (por eso su bit P debe ser siempre igual a uno), pero tanto las tablas PD como las PT pueden no estar presentes en ella.
- d) ...en un mismo sistema no pueden convivir en ningún caso páginas de 4MB con páginas de 2MB.

2) En la paginación puede afirmarse que...

- a) ...el tamaño de todas las tablas que se usan en todos los tipos de paginación es de 4KB, y por lo tanto, todas ellas pueden tratarse a su vez como páginas de tamaño no extendido.
- b) ...en un mismo sistema no pueden convivir en ningún caso páginas de 4KB con páginas de 2MB.
- c) ... si la paginación no está activada, el margen de direcciones físicas no es nunca mayor de 4GB.
- d) Ninguna de las anteriores es cierta.

3) Un programa con CPL=2 tiene que incrementar una variable de tamaño byte, para lo cual realiza 2 accesos consecutivos: lee la variable y la escribe (una vez incrementada). Suponiendo que previamente la traducción de página lineal a física no estuviera en la TLB y que la página tuviera en las tablas el bit D a cero, ¿en cuales de estos accesos podría usarse la traducción del TLB sin necesidad de acudir a las tablas?.

- a) En ninguno de los dos.
- b) En ambos.
- c) En el segundo.
- d) Depende del bit G.

⁸ En el examen final había sólo cinco de estas diez preguntas (las cuestiones 1, 3, 6, 8 y 10).



Problema 4 final/3 parcial. (Septiembre tarde).

(Cont).

- 4) Respecto a los parámetros PCD y PWT que aparecen en las entradas de las tablas de paginación, se puede decir que ...
- a) ... representan el tratamiento que debe hacer la caché de las tablas y/o de las páginas definidas en el proceso de paginación.
 - b) ... definen los derechos de diferentes tipos de usuarios sobre las páginas.
 - c) ... indican si la traducción de direcciones en la TLB debe borrarse o no automáticamente cada vez que se produzca un cambio de tarea.
 - d) ... ninguna de las respuestas anteriores es cierta.
- 5) En paginación con direccionamiento extendido.
- a) El bit PS no tiene ningún significado.
 - b) El tamaño de las entradas de las tablas es de 32 Bytes.
 - c) No se admiten páginas de 4KB.
 - d) No todas las tablas tienen un tamaño de 4KB.
- 6) En paginación, la dirección lineal ...
- a) ... tiene 32 bits con direccionamiento no extendido y 64 bits con direccionamiento extendido.
 - b) ... tiene 32 bits con direccionamiento no extendido y 36 bits con direccionamiento extendido..
 - c) ... tiene siempre 32 bits.
 - d) ... tiene siempre 36 bits.
- 7) En paginación, la dirección lineal y la dirección física ...
- a) ... tienen en común el offset.
 - b) ... se diferencian en el offset.
 - c) ... tienen siempre el mismo tamaño.
 - d) ... tienen siempre tamaños distintos.
- 8) En paginación con direccionamiento extendido.
- a) Si PS=1, el tamaño de la página es de 4KB.
 - b) Si PS=1 no hay Tabla de Páginas (PT) para ese acceso.
 - c) Hay accesos en los que se usan cuatro tablas para llegar a la página.
 - d) Hay accesos en los que sólo se usa una tabla para llegar a la página.

