A **Apellidos** _____ **Nombre** _____ **DNI** _____

Examen de Arquitectura de Computadores (Telemática) **Convocatoria de Julio: 9 – 07 – 2010**

El siguiente código incompleto, trata de averiguar si existe un fichero cuya ruta se proporciona. En caso de que exista, se muestra por pantalla el nombre (sin la ruta) del fichero y su tamaño expresado en bytes, siempre que éste no sea superior a MAXDWORD bytes (ver anexo). Si el tamaño del fichero es superior a MAXDWORD bytes, sólo se mostrará por pantalla que el tamaño del fichero es muy grande. Esta búsqueda se hace mediante la función de WIN32 *FindFirstFile*. Si la función no consigue encontrar el fichero indicado se debe indicar este hecho por pantalla. Si la función fallase por otro motivo, se ha de mostrar un mensaje de error.

```
#include <windows.h>
#include <conio.h>
#include <stdio.h>

int main()
{
   WIN32_FIND_DATA FindFileData;
   HANDLE h;
   DWORD err;
   char ruta[]="c:\\SuperMario.jpg";

   //Utilizar la función FindFirstFile para buscar el fichero
   //cuya ruta se especifica en la variable ruta

   h = FindFirstFile(ruta, &FindFileData);                    0,5

   //Si la llamada no ha tenido éxito informar al usuario.
   //Indicar si ha fallado por no encontrar el fichero o por otro
   //motivo
   if (h==INVALID_HANDLE_VALUE)                               0,5
   {
        err=GetLastError();

        //Si el fichero no fue encontrado…
        if (err==ERROR_FILE_NOT_FOUND)                        0,5
             printf("Fichero no encontrado\n");

        else //Falló por otro motivo
             printf("Error al ejecutar la función FindFirstFile\n");

        getch();
      return 0;
   }

   //Si se ha llegado hasta aquí, es que la llamada ha tenido éxito.
   //Informar al usuario de que se encontró el fichero, indicando su
   //nombre (SIN la ruta de directorios, sólo el nombre)
   printf ("Se ha encontrado el fichero %s\n",                0,5
            FindFileData.cFileName);

   //Mostrar por pantalla el tamaño del fichero si éste
   //es inferior a MAXDWORD bytes. En caso contrario, mostrar
   //un mensaje indicando que el fichero es muy grande.

   if (FindFileData.nFileSizeHigh!=0)                         1
      printf("El tamaño del fichero es superior a %d bytes\n",
              MAXDWORD);
   else
         printf("El tamaño del fichero es %d\n",
                 FindFileData.nFileSizeLow);

   getch();
   return 0;
}
```

Se dispone de un computador con las siguientes características:

- Direcciones virtuales de 32 bits.
- Direcciones físicas de 20 bits.
- Páginas de 2KB.
- Cada posición de memoria almacena 1 byte.

El computador tiene instalado un módulo de memoria física de 64KB. Conociendo esta información, responde a las siguientes preguntas:

— ¿Cuál es la dirección física más significativa que puede ser utilizada? Contestar en hexadecimal.

0FFFF **0,5**

A

Un usuario de este computador ha iniciado la ejecución de una aplicación. En un determinado instante durante su ejecución, el estado de su tabla de páginas es el indicado a continuación:

Tabla de Páginas

Nº Pag. Virtual	Presencia	Usuario / Supervisor	Read /Write	Nº Pag. Fis. / Offset Dis.
02 00 00	SI	Usuario	Read	00A
02 00 01	SI	Usuario	Read	00B
02 00 02	NO	Usuario	Read	Offset X
02 00 03	NO	Usuario	Read	Offset Y
04 AA 00	NO	Usuario	Read-Write	Offset Z
04 AA 01	SI	Usuario	Read-Write	007
04 AA 02	NO	Usuario	Read-Write	Offset N
1F FF FD	SI	Supervisor	Read-Write	01D
1F FF FE	SI	Supervisor	Read-Write	01E
1F FF FF	SI	Supervisor	Read-Write	01F

— Escribe la dirección virtual menos significativa que pueda ser accedida por esta aplicación para escritura sin generar una excepción. Contestar en hexadecimal.

25 50 08 00 — **0,5**

— Escribe la dirección física que le corresponde a la dirección virtual de la pregunta anterior. Contestar en hexadecimal.

0 38 00 — **0,25**

— La página virtual 04 AA 02h es la única página dedicada a contener sección de datos, el cual se compone de un array de 512 enteros (cada entero ocupa 4 bytes). ¿En qué rango de direcciones virtuales se mapea el elemento con índice 10 de dicho vector? Contestar en hexadecimal.

25 50 10 28 – 25 50 10 2B — **0,75**

A continuación se muestra un programa en C que calcula la suma de los *n* primeros números mediante una función iterativa.

```
#include <stdio.h>

int sumaN (int n){

       int resultado, contador;

       resultado=0;
       contador=n;

       while (contador!=0){
              resultado=resultado+contador;
              contador=contador-1;
       }

       return resultado;

}

int main()
{

       printf("Sumatorio de 5 primeros números: %d\n", sumaN(5));

   return 0;
}
```

— Escribe a continuación las sentencias ensamblador que se generarían al traducir el bucle *while* completo.

Nota: No se pueden utilizar etiquetas de parámetros de funciones ni de variables locales.

1

```
bucle:

  cmp DWORD PTR [ebp-8], 0
  je  final
  mov eax, [ebp-8]
  add eax, [ebp-4]
  mov [ebp-4], eax
  mov eax, [ebp-8]
  dec eax
  mov [ebp-8], eax
  jmp bucle
final:
```

A **Apellidos** _____ **Nombre** _____ **DNI** _____

Examen de Arquitectura de Computadores (Telemática) **Convocatoria de Julio: 9 – 07 – 2010**

A continuación se muestra el código de un programa escrito en ensamblador de Intel. En dicho programa se ha creado un procedimiento encargado de concatenar cadenas de caracteres llamada *concatena*. Este procedimiento recibe dos parámetros **por referencia** en el siguiente orden:

1. Dirección de la cadena destino.
2. Dirección de la cadena origen.

Este procedimiento comienza con un bucle que busca el primer carácter ASCII 0 de la cadena destino. Una vez encontrado, mediante otro bucle, copia los caracteres de la cadena origen a partir de dicha posición de la cadena destino. Este bucle finaliza al encontrar un ASCII 0 en la cadena origen.

La cadena origen no sufre ninguna modificación tras la ejecución del procedimiento, mientras que la cadena destino queda modificada, habiéndosele añadido la cadena origen al final. El procedimiento no comprueba que la cadena destino sea lo suficientemente grande como para albergar la cadena concatenada, por lo que este aspecto es responsabilidad del programador. El procedimiento no devuelve ningún valor.

```
.386
.MODEL FLAT, stdcall

ExitProcess PROTO, :DWORD

.DATA
  cadena1 DB "Vamonos, atomos.", 0
  cadena2 DB "Venga, vamonos.", 0

  destino DB 0 DUP (64)


.CODE

concatena PROC
      push ebp
      mov  ebp, esp

      push esi
      push edi

      ;Lectura de parámetros
      (--2--)                  ;EDI=Dir. cadena destino
                               ;ESI=Dir. cadena origen

bucle:
      cmp  BYTE PTR [edi], 0
      je seguir
      inc edi
      jmp bucle
seguir:

bucle2:
      cmp BYTE PTR [esi], 0
      je final
      (--3--)


final:
      pop edi
      pop esi

      pop ebp
      ret 8
concatena ENDP


inicio:

      (--1--)


      push 0
      call ExitProcess

END inicio
```

Escribe las instrucciones que consideres necesarias en el programa principal, hueco (--1--), para que la variable global *destino* contenga la cadena "*Vamonos, atomos. Venga, vamonos.*"

0,5

```
push OFFSET destino
push OFFSET cadena1
call concatena


push OFFSET destino
push OFFSET cadena2
call concatena
```

A

— Escribe las instrucciones necesarias en el hueco (--**2**--) para realizar la lectura de los parámetros como se indica en el comentario.

0,5

```
mov  edi, [ebp+12]
mov  esi, [ebp+8]
```

— Escribe las instrucciones correspondientes al hueco (--**3**--) para completar el bucle encargado de copiar los caracteres de la cadena origen a la cadena destino.

0,5

```
 mov  al, [esi]
 mov  [edi], al
 inc  edi
 inc  esi
 jmp  bucle2
```

— Contesta a las siguientes preguntas breves:

1

Escribe las instrucciones que consideres oportunas para convertir un dato de 8 bits interpretado con signo en un dato de 32 bits. El dato origen se encuentra en el registro BL. El dato convertido debe dejarse en el registro EDX.

```
movsx  edx, bl
```

Escribe una sola instrucción lógica que convierta una letra mayúscula almacenada en el registro AL en minúscula. Datos: ASCII('A') = 41h; ASCII('a') = 61h

```
or al, 20h
```

Escribe una secuencia de instrucciones que lean un dato del puerto de E/S 300h y lo almacenen en la variable *var*, de tipo byte, definida en la sección de datos.

```
mov dx, 300h
in  al, dx
mov [var], al
```

Indica para qué se utilizan las excepciones de tipo fallo en la arquitectura IA-32.

Se utilizan para señalizar errores no catastróficos, que pueden ser tratados sin que se pierda la estabilidad del sistema.

— Define los conceptos de arquitectura y organización de computadores.

0,5

Arquitectura:

Especificación del computador en su nivel de lenguaje máquina. Es decir, el juego de instrucciones, los tipos de operandos sobre los que éstas actúan y el espacio o espacios de direcciones.

Organización:

Conjunto de componentes físicos que conforman el ordenador, así como sus interrelaciones.

— Responde a las siguientes preguntas.

0,5

¿Por qué en un sistema operativo moderno se puede ejecutar un programa que sea más grande que la memoria física?

Porque los sistemas operativos modernos permiten utilizar parte del disco duro como una extensión de la memoria física, lo que nos permite cargar programas más grandes que la propia memoria.

En un sistema de memoria virtual paginada, ¿cómo se evita que un proceso no escriba en posiciones físicas asignadas a otro proceso?

Utilizando una tabla de páginas distinta para cada proceso, en la cual sólo se mapean las páginas propias de cada proceso (además de ciertas partes del SO).

— Codifica la siguiente instrucción: `mov DWORD PTR [esi+128], 1025`
Contestar en hexadecimal.

0,5

```
C7 86 80 00 00 00 01 04 00 00
```

FindFirstFile

Busca en un directorio por un fichero o subdirectorio cuyo nombre coincida con uno especificado (por defecto de forma completa; de forma parcial si se utilizan comodines).

```
HANDLE FindFirstFile (

  LPCTSTR lpFileName,

  LPWIN32_FIND_DATA lpFindFileData

);
```

Parámetros

lpFileName
[in] Ruta del fichero o directorio a buscar. Puede incluir comodines, como por ejemplo, un asterisco (*) o un signo de interrogación (?).
Este parámetro no debe ser NULL, ni una cadena inválida (como podría ser una cadena vacía). Si esta cadena termina con un comodín, un punto (.) o el nombre de un directorio, el usuario debe tener permisos para acceder a dicho directorio y a todos sus subdirectorios.

lpFindFileData
[out] Puntero a una estructura **WIN32_FIND_DATA**, que recibe información sobre el fichero o directorio encontrado.

Valores de retorno
Si la función tiene éxito, el valor devuelto es el manejador que se utilizará en las subsiguientes llamadas a **FindNextFile** o **FindClose**, y el parámetro *lpFindFileData* contendrá información sobre el primer fichero o directorio encontrado.
Si la función falla o no puede encontrar ningún fichero cuyo nombre coincida con el parámetro *lpFileName*, el valor devuelto es **INVALID_HANDLE_VALUE** y el contenido de *lpFindFileData* es indeterminado. Para obtener más información sobre el error, se ha de llamar a la función **GetLastError**.
Si la función falla por no haber encontrado ningún fichero con el nombre indicado, la función **GetLastError** devuelve el valor **ERROR_FILE_NOT_FOUND**.

GetLastError

La función **GetLastError** devuelve el código asociado al último error producido en el hilo llamador. El código del último error se mantiene para cada hilo en una tabla. Ningún hilo sobrescribe el valor del último código de error de ningún otro hilo.

```
DWORD GetLastError(void);
```

Parametros

Esta función no recibe parámetros.

Valores de retorno

El valor de retorno es el código del último error que se produjo en el hilo que invoca la función. Las funciones actualizan este valor mediante una llamada a la función **SetLastError**.

WIN32_FIND_DATA

La estructura **WIN32_FIND_DATA** contiene información sobre el fichero encontrado por la función **FindFirstFile**, **FindFirstFileEx** o **FindNextFile**.

```
typedef struct _WIN32_FIND_DATA {
  DWORD    dwFileAttributes;
  FILETIME ftCreationTime;
  FILETIME ftLastAccessTime;
  FILETIME ftLastWriteTime;
  DWORD    nFileSizeHigh;
  DWORD    nFileSizeLow;
  DWORD    dwReserved0;
  DWORD    dwReserved1;
  TCHAR    cFileName[MAX_PATH];
  TCHAR    cAlternateFileName[14];
} WIN32_FIND_DATA, *PWIN32_FIND_DATA, *LPWIN32_FIND_DATA;
```

Miembros

dwFileAttributes
Atributos del fichero.

ftCreationTime
Estructura de tipo **FILETIME** que especifica cuando se ha creado el fichero. Si el sistema de ficheros utilizado no da soporte al instante de creación de fichero, este miembro es 0 (cero).

ftLastAccessTime
Estructura de tipo **FILETIME**. Para un fichero, la estructura especifica el último instante en el que ha sido leído o escrito. Para un directorio, la estructura especifica cuando se ha creado. Tanto para ficheros como para directorios, la fecha especificada es correcta, pero la hora siempre se fija en media noche. Si el sistema de ficheros utilizado no da soporte al instante del último acceso, este miembro es 0 (cero).

ftLastWriteTime
Estructura de tipo **FILETIME**. Para un fichero, la estructura especifica el último instante en el que ha sido escrito. Para un directorio, la estructura especifica cuando se ha creado. Si el sistema de ficheros utilizado no da soporte al instante del último acceso, este miembro es 0 (cero).

nFileSizeHigh
La parte alta del tamaño del fichero, en bytes. Este valor es 0 (cero) salvo que el tamaño del fichero sea superior a MAXDWORD bytes. Si el tamaño es superior a MAXDWORD bytes, el tamaño del fichero es igual a (*nFileSizeHigh* * (MAXDWORD)) + *nFileSizeLow*.

nFileSizeLow
La parte baja del tamaño del fichero, en bytes.

dwReserved0
Si el miembro *dwFileAttributes* incluye el atributo **FILE_ATTRIBUTE_REPARSE_POINT**, este miembro especifica el punto de reanálisis. En otro caso, este valor es indefinido y no debe utilizarse.

dwReserved1
Reservado para uso futuro.

cFileName
Nombre del fichero.

cAlternateFileName
Nombre alternativo del fichero.