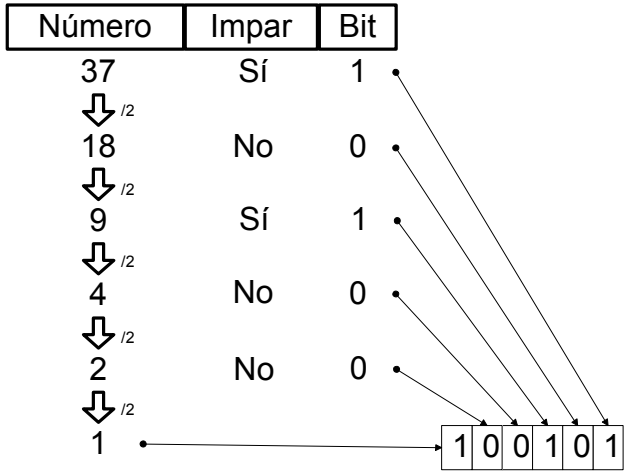
A **Apellidos** _____ **Nombre** _____ **DNI**_____

Examen de Arquitectura de Computadores (Telemática) **Convocatoria de Junio: 25 – 05 – 2011**

Se dispone de un programa escrito en ensamblador encargado de transformar un número escrito en decimal a binario. El número binario obtenido al realizar la transformación se guardará en una cadena de texto, en la cual cada carácter representa un bit. El método a utilizar para realizar la transformación es el ya conocido método de las divisiones por 2 sucesivas. Partiendo de un número *n*, éste se va dividiendo sucesivamente (división entera) entre 2 hasta que el cociente sea 1 ó 0, momento en el que el algoritmo finaliza. El resto de cada una de esas divisiones proporcionará un nuevo bit al número en binario. Los bits obtenidos de este modo se completan de derecha a izquierda, es decir, el primer bit obtenido es el bit menos significativo del número en binario. Para variar "un poco" el modo tradicional, en lugar de consultar el resto de las divisiones, se comprobará si el cociente de cada división es par o impar (en realidad es lo mismo: si al dividir un número por 2 el cociente es 0, éste es par, y si es 1 es impar). Por ejemplo, si queremos transformar el número 37 a binario, comenzaremos con poner un bit '1' en la posición menos significativa del resultado. Hacemos la división entera por 2 y vamos colocando 1 ó 0 según si el número obtenido es par o impar. Cuando el resultado de dividir sea 1 ó 0, éste se coloca en la posición más significativa y el algoritmo termina.



Para realizar esta tarea, se cuenta con tres procedimientos que se describen a continuación:

- Procedimiento **DividePor2**: Divide entre 2 el número que se le pasa como parámetro y devuelve el cociente en el registro EAX (división entera). El parámetro se pasa POR VALOR y ha de ser de 32 bits.
- Procedimiento **EsImpar**: Determina si el número que se le pasa como parámetro es Par o Impar. La función devuelve en el registro EAX un valor **1** si el parámetro es impar y **0** si es par. El parámetro se pasa POR VALOR y ha de ser de 32 bits.
- Procedimiento Transforma: Es el procedimiento encargado de transformar un valor codificado en decimal a binario. El número transformado a binario se escribe en una cadena de caracteres, en la que cada carácter representa un bit. Recibe tres parámetros en el siguiente orden de apilación:

  1. Número a transformar: Debe de estar codificado en decimal. Se pasa POR VALOR y ha de ser de 32 bits.
  2. Cadena destino: Cadena de caracteres (elementos de tipo BYTE) donde se escribirá el número en binario. Se pasa POR REFERENCIA. Todas sus posiciones deben estar inicializadas con el carácter ASCII del número cero, '0'.
  3. Tamaño de la cadena: Se pasa POR VALOR y ha de ser de 32 bits.

  El procedimiento carga en el registro EBX el número a transformar, en el registro ESI la dirección de la cadena donde se escribirá el resultado y en ECX el tamaño de ésta. A continuación, sumará a ESI (tamaño – 1) para que éste pase a contener la dirección de la última posición de la cadena (puesto que el número en binario lo comenzamos a calcular por su bit menos significativo). Una vez hecho esto comienza el bucle que realiza el cálculo. Este bucle, controlado por condición, termina cuando el contenido del registro EBX es menor o igual que 1. En cada iteración se ha de comprobar si el contenido de EBX es par o impar utilizando el procedimiento **EsImpar**. En caso de ser impar, se escribe un ASCII '1' en la posición de la cadena apuntada por ESI, y si es par no se hace nada (puesto que la cadena está inicializada con ceros). A continuación, se divide el contenido del registro EBX por 2 utilizando el procedimiento **DividePor2** y se actualiza el registro ESI. Finalmente, se pone un ASCII '1' si el número que quedó en EBX es 1 y '0' en caso de que haya quedado un 0. Este procedimiento no devuelve ningún valor.

  Por simplicidad, el código del procedimiento no tiene en cuenta los casos en los que la codificación en binario del número a transformar requiera un número de bits superior al número de BYTES de la cadena destino.

  Nota: Para escribir el código ASCII del número 0 en una posición de la cadena, se ha de poner el 0 entre comillas simples ('0') o bien el código ASCII del 0 (48). Análogamente, el código ASCII del número 1 se ha de poner como '1' ó 49.

A

```
.386
.MODEL FLAT, stdcall

ExitProcess PROTO, :DWORD

.DATA

 numero        DD 37
 NumBinario    DB "00000000"
 tamanio       DD 8



.CODE

DividePor2 PROC
       push ebp
       mov  ebp, esp

       mov  eax, [ebp+8]
       shr  eax, 1 ;Desplaza los bits una posición a la derecha

       pop  ebp
       ret  4
DividePor2 ENDP


EsImpar PROC
       push ebp
       mov  ebp, esp

       push ebx
       mov  ebx, [ebp+8]  ; Dejar en EBX el valor del parámetro
       xor  eax, eax

       (--1--)      ;Determinar si EBX es PAR o IMPAR
                    ;Si es Impar: EAX=1, si es Par: EAX=0
       pop ebx
       pop ebp
       ret 4
EsImpar ENDP
```

```
Transforma PROC
       push ebp
       mov  ebp, esp

       push ebx
       push ecx
       push esi

       mov  ebx, [ebp+16] ;numero
♠♠♠    mov  esi, [ebp+12] ;dir.
       mov  ecx, [ebp+8]  ;tamanio

       add  esi, ecx
       dec  esi            ;Poner esi en la última posición

bucle:
       cmp  ebx, 1
       jbe  finBucle

       (--2--)      ;Poner '1' en la posición actual de la cadena
                    ;si EBX es IMPAR y 0 si es PAR. Dividir EBX por 2
                    ;y actualizar el contenido de ESI para dejarlo
                    ;listo para la siguiente iteración

       jmp  bucle

finBucle:

       (--3--)

       pop  esi
       pop  ecx
       pop  ebx
       pop  ebp
       ret  12
Transforma ENDP

inicio:
       (--4--)

       push 0
       call ExitProcess
END inicio
```

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

Examen de Arquitectura de Computadores (Telemática) **Convocatoria de Junio: 25 – 05 – 2011**

— Escribe las instrucciones necesarias del hueco (--**1**--) para determinar si el contenido del registro EBX es un número par o un número impar. Esto debe hacerse comprobando si el bit menos significativo del número es 1 ó 0 mediante una función lógica y una máscara. Si es 1, el número es IMPAR y si es 0 es PAR. Si el número era IMPAR, dejar un 1 en EAX y no hacer nada en caso contrario (puesto que EAX está inicializado a 0).

0,5

```
                              También vale:
  and  ebx, 1                 and  ebx, 1
  cmp  ebx, 0                 mov  eax, ebx
  je   finFuncion
  mov  eax, 1
finFuncion:
```

— Escribe las instrucciones necesarias del hueco (--**2**--) para completar el bucle del procedimiento *Transforma*. En cada iteración del bucle se debe comprobar si el contenido de ebx es impar. Si lo es, poner un '1' en la posición correspondiente de la cadena, actualizar el puntero (registro esi) y dividir por 2 el contenido del registro ebx.

1

```
  push ebx
  call EsImpar
  cmp  eax, 1
  jne  seguir
  mov  BYTE PTR [esi], '1'
seguir:
  dec  esi
  push ebx
  call DividePor2
  mov  ebx, eax
```

— Escribe las instrucciones necesarias del hueco (--**3**--) para poner un '1' ó un '0' en la posición correspondiente de la cadena si el valor final de ebx es 1 ó 0 respectivamente.

0,75

```
                              También vale:
  cmp  ebx, 0
  je   PonerCero                 cmp ebx, 1
  mov  BYTE PTR [esi], '1'       jne final
  jmp  final                     mov BTE PTR [esi], '1'
PonerCero:                    final:
  mov  BYTE PTR [esi], '0'
final:
```

— Escribe las instrucciones necesarias del hueco (--**4**--) correspondiente al programa principal de la aplicación, donde se ha de llamar al procedimiento *Transforma* para convertir a binario el número guardado en la variable global *numero*. El número convertido debe quedar escrito en la variable global de tipo cadena de caracteres *NumBinario* cuyo tamaño se especifica en la variable global *tamanio*.

0,5

```
push [numero]
push OFFSET NumBinario
push [tamanio]
call Transforma
```

— Codifica la instrucción: `mov esi, [ebp+12]`, señalada en el código con el símbolo ♠♠♠. Contesta en hexadecimal.

0,5

```
8B 75 0C
```

— Contesta las siguientes preguntas breves:

1

Indica el cometido de la rutina del sistema operativo que realiza el tratamiento del fallo de página.

Copiar la página a la que se hecho referencia del disco en un marco de página de la memoria física y actualizar la tabla de páginas para que refleje la nueva ubicación de la página.

¿Qué indica el bit R/W de la tabla de páginas, y cómo se utiliza?

Indica si la página correspondiente es de sólo lectura o de lectura/escritura.

Se utiliza marcando como de sólo lectura las páginas de código y como de lectura/escritura las páginas de datos del programa correspondiente.

¿Cuáles son los cuatro componentes típicos del núcleo de un sistema operativo moderno?

Gestor de procesos, gestor de memoria, gestor de ficheros y gestor de entrada-salida.

A

Responde brevemente a las siguientes cuestiones:

¿Cómo se llama el registro utilizado para almacenar los bits de estado y control de la CPU en la arquitectura IA32? 1

EFLAGS

¿Qué tipos de excepciones existen en la arquitecuta IA32?

Fallos, abortos y trampas

¿Qué información se almacena en el marco de pila de una función?

Sus parámetros, dirección de retorno, valor anterior del registro EBP y variables locales.

A continuación se muestra un programa (incompleto) escrito en C cuyo objetivo es determinar si el sistema está actualmente funcionando con el horario de verano o el horario de invierno, y la fecha en la que se realizará el siguiente cambio de hora. Tras mostrar esta información, el usuario modificará la fecha en la que se realizará dicho cambio. Para ello se utilizarán las funciones del API de WIN32 **GetTimeZoneInformation** y **SetTimeZoneInformation.** Dichas funciones trabajan con una estructura de tipo **TIME_ZONE_INFORMATION**. La especificación de estas funciones y de la estructura se encuentra en el anexo. A continuación se muestra un ejemplo de la salida que tiene que tener el programa. Téngase en cuenta a la hora de completar las sentencias **printf**.

```
Actualmente estamos utilizando: Hora estándar romance
El cambio al horario de verano sera la semana 5 del mes 3 a las 2
La funcion SetTimeZoneInformation funciono
```

Nota: Según se especifica en el anexo, como semana 5 se entiende la última semana de dicho mes, aunque el día del cambio se corresponda con la cuarta semana.

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

int main()

{
	//Declaración de variables

	TIME_ZONE_INFORMATION tzi;
	DWORD timeZone;
	BOOL resultado;

	//Obtener información de la configuración horaria
	//mediante GetTimeZoneInformation
	timeZone=GetTimeZoneInformation(&tzi);                    0,5

	//Si estamos en horario de invierno…
	if (  timeZone==TIME_ZONE_ID_STANDARD                     )
	{
	 printf("Actualmente estamos utilizando: %s\n",
	       tzi.StandardName
	 );

	 printf("El cambio al horario de verano sera la semana
	      %d del mes %d a las %d\n", tzi.DaylightDate.wDay,
	      tzi.DaylightDate.wMonth, tzi.DaylightDate.wHour
	);
	}
	//Si estamos en horario de verano…                        1
	else if (timeZone==TIME_ZONE_ID_DAYLIGHT                  )
	{
	      printf("Actualmente estamos utilizando: %s\n",
	            tzi.DaylightName
	      );

	      printf("El cambio al horario de invierno sera la semana
	            %d del mes %d a las %d\n", tzi.StandardDate.wDay,
	           tzi.StandardDate.wMonth, tzi.StandardDate.wHour
	      );
	}
```

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

Examen de Arquitectura de Computadores (Telemática) **Convocatoria de Junio: 25 – 05 – 2011**

```
        //Cambiar el día en que empezará el horario de verano
        //en la estructura tzi
        tzi.DaylightDate.wDay=27;

        //Hacer efectivo el cambio utilizando SetTimeZoneInformation
        resultado=SetTimeZoneInformation(&tzi);          0,5

        //Comprobar si la función se ejecutó correctamente
        if (  resultado==0                               0,25  )
                printf("La funcion SetTimeZoneInformation fallo\n");
        else
                printf("La funcion SetTimeZoneInformation funciono\n");

        _getch();

    return 0;
}
```

Un computador utiliza un sistema de memoria virtual con las siguientes características:

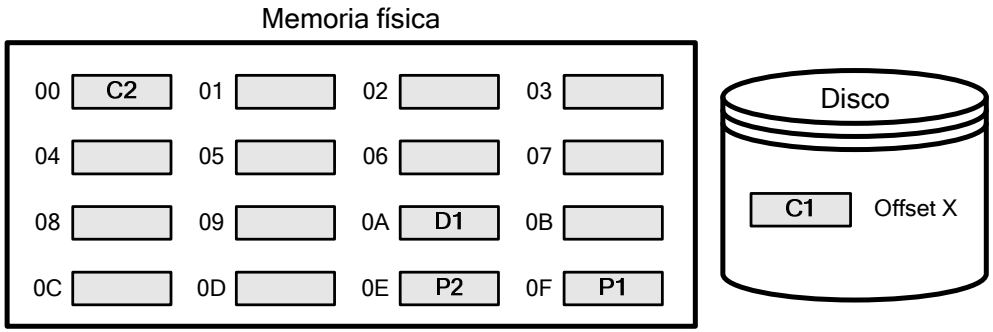
Direcciones virtuales: 16 bits

Direcciones físicas: 12 bits

Tamaño de página: 64 bytes (cada posición de memoria alberga un byte).

Se está ejecutando en este computador un proceso que consta de 1 página para la sección de datos (D1), 2 páginas para la sección de pila (P1 y P2) y 2 páginas para el código (C1, C2). A continuación se muestra la ubicación física de cada una de estas páginas en memoria física y el fichero de paginación del disco.

Además, se sabe que la sección de código comienza siempre a partir de la dirección virtual **2000h**, la sección de datos comienza siempre a partir de la dirección virtual **5000h**, y la pila a partir de la dirección virtual **FFFFh** (la pila crece hacia direcciones menos significativas).



— Según esta información, completa la tabla de páginas mostrada a continuación. Rellena sólo las entradas correspondientes a páginas utilizadas por el programa en ejecución.

Tabla de Páginas

Nº Pag. Virtual (Hex)	Pre-sencia	Nº Pag. Fis. (Hex) / Offset Dis.
080	No	Offset X
081	Si	00
140	Si	0A
3FF	Si	0F
3FE	Si	0E

1

— Determina el % del espacio de direcciones físico ocupado por la memoria física.

25% **0,5**

— En la sección de datos de este programa hay declarado un vector de 10 enteros (cada entero ocupa 4 bytes) y a continuación una variable numérica de tipo entero. Al haber sido declarado antes, el vector ocupará un conjunto de direcciones virtuales menos significativas que la variable. Determina el rango de direcciones virtuales ocupadas por dicha variable. Contestar en hexadecimal.

50 28 – 50 2B **0,5**

— Determina el rango de direcciones físicas ocupadas por esa misma variable. Contestar en hexadecimal.

2 A8 – 2 AB **0,5**

GetTimeZoneInformation

Devuelve la configuración actual de la zona horaria. Esta configuración controla las diferencias entre el Tiempo Universal Coordinado (UTC) y la hora local.

```
DWORD GetTimeZoneInformation(
  LPTIME_ZONE_INFORMATION lpTimeZoneInformation
);
```

Parámetros

lpTimeZoneInformation
[out] Puntero a una estructura de tipo **TIME_ZONE_INFORMATION** que recibirá la configuración actual.

Valores de retorno

Si la función tiene éxito, devolverá uno de los siguientes valores:

- TIME_ZONE_ID_UNKNOWN: No se utiliza el cambio de horario de verano a horario de invierno al no estar especificadas las fechas o al estar desactivado el ajuste de horario automático.
- TIME_ZONE_ID_STANDARD: El sistema está utilizando el horario de invierno.
- TIME_ZONE_ID_DAYLIGHT: El sistema está utilizando el horario de verano.
- TIME_ZONE_ID_INVALID: La llamada a la función falló.

Observaciones

Todas las traslaciones entre la hora UTC y la hora local se basan en la siguiente igualdad:
UTC = hora local + desplazamiento
El desplazamiento es la diferencia, en minutos, entre la hora UTC y la hora local.

SetTimeZoneInformation

Establece la configuración horaria a utilizar. Esta configuración controla las diferencias entre el Tiempo Universal Coordinado (UTC) y la hora local.

```
BOOL SetTimeZoneInformation(
  LPTIME_ZONE_INFORMATION lpTimeZoneInformation
);
```

Parámetros

lpTimeZoneInformation
[in] Puntero a la estructura de tipo **TIME_ZONE_INFORMATION** que contiene la nueva configuración.

Valores de retorno

Si la función tiene éxito, devuelve un valor distinto de cero. Si la función falla, devuelve cero.

TIME_ZONE_INFORMATION

La estructura **TIME_ZONE_INFORMATION** especifica la configuración del huso horario.

```
typedef struct _BY_HANDLE_FILE_INFORMATION {
  LONG       Bias;
  WCHAR      StandardName[32];
  SYSTEMTIME StandardDate;
  LONG       StandardBias;
  WCHAR      DaylightName[32];
  SYSTEMTIME DaylightDate;
  LONG       DaylightBias;
} TIME_ZONE_INFORMATION, *PTIME_ZONE_INFORMATION;
```

Miembros

Bias

El desplazamiento utilizado actualmente para realizar la traslación horaria entre la hora UTC y la hora local. El desplazamiento es la diferencia, en minutos, entre la hora UTC y la hora local. Todas las traslaciones entre la hora UTC y la hora local se basan en la siguiente igualdad:
UTC = hora local + desplazamiento
Este miembro es obligatorio.

StandardName

Cadena de caracteres que contiene una descripción del horario estándar (horario de invierno) utilizado. Por ejemplo, puede indicar "Hora estándar romance", que indicaría que se está utilizando el horario de invierno europeo. Esta cadena puede estar vacía.

StandardDate

Estructura de tipo **SYSTEMTIME** que contiene la fecha y la hora (local) en la que se cambiará de horario de verano a horario de invierno. Si el huso horario no soporta el cambio a horario de verano o si el llamador necesita desactivarlo, el miembro **wMonth** en la estructura **SYSTEMTIME** debe ser cero. Si se especifica esta fecha, el miembro **DaylightDate** de esta estructura debe de estar también especificado. En caso contrario, el sistema asumirá que esta información es inválida y los cambios no serán aplicados.

Para seleccionar el día del mes en que se producirá el cambio de hora, establecer el miembro **wYear** a cero, los miembros **wHour** y **wMinute** con la hora y el minuto en que se realizará el cambio, el miembro **wDayOfWeek** al día apropiado de la semana y el miembro **wDay** indica el número de aparición de ese día de la semana dentro de su mes (de 1 a 5, donde 5 indica la última aparición en ese mes, aunque ese día de la semana no aparezca 5 veces).

Utilizando esta notación, especificar las 02:00 del primer domingo de Abril sería del siguiente modo: **wHour** = 2, **wMonth** = 4, **wDayOfWeek** = 0, **wDay** = 1. Especificar las 02:00 del último jueves de Octubre sería del siguiente modo: **wHour** = 2, **wMonth** = 10, **wDayOfWeek** = 4, **wDay** = 5.

Si el miembro **wYear** es distinto de cero, la fecha de transición es absoluta, es decir, el cambio de hora sólo ocurrirá una vez. En otro caso, es una fecha relativa, y ocurrirá anualmente.

StandardBias

El valor de desplazamiento que se utilizará (respecto a la hora UTC) durante el horario de invierno. Este miembro será ignorado si el valor de **StandardDate** no se proporciona.
Este valor será sumado al valor del miembro **Bias** para formar el desplazamiento utilizado en el horario de invierno. En la mayoría de los husos horarios, el valor de este miembro es cero.

DaylightName

Cadena de caracteres que contiene una descripción del horario de verano utilizado. Por ejemplo, puede indicar "Hora de verano romance", que indicaría que se está utilizando el horario de verano europeo. Esta cadena puede estar vacía.

DaylightDate

Estructura de tipo **SYSTEMTIME** que contiene la fecha y la hora (local) en la que se cambiará de horario de invierno a horario de verano. Si el huso horario no soporta el cambio a horario de verano o si el llamador necesita desactivarlo, el miembro **wMonth** en la estructura **SYSTEMTIME** debe ser cero. Si se especifica esta fecha, el miembro **StandardDate** de esta estructura debe de estar también especificado. En caso contrario, el sistema asumirá que esta información es inválida y los cambios no serán aplicados.

Para seleccionar el día del mes en que se producirá el cambio de hora, establecer el miembro **wYear** a cero, los miembros **wHour** y **wMinute** con la hora y el minuto en que se realizará el cambio, el miembro **wDayOfWeek** al día apropiado de la semana y el miembro **wDay** indica el número de aparición de ese día de la semana dentro de su mes (de 1 a 5, donde 5 indica la última aparición en ese mes, aunque ese día de la semana no aparezca 5 veces).

Si el miembro **wYear** es distinto de cero, la fecha de transición es absoluta, es decir, el cambio de hora sólo ocurrirá una vez. En otro caso, es una fecha relativa, y ocurrirá anualmente.

DaylightBias

El valor de desplazamiento que se utilizará (respecto a la hora UTC) durante el horario de verano. Este miembro será ignorado si el valor de **DaylightDate** no se proporciona.
Este valor será sumado al valor del miembro **Bias** para formar el desplazamiento utilizado en el horario de invierno. En la mayoría de los husos horarios, el valor de este miembro es -60.

SYSTEMTIME

La estructura **SYSTEMTIME** representa una fecha y una hora utilizando miembros individuales para el mes, día, año, día de la semana, hora, minuto, segundo y milisegundo.

```
typedef struct _SYSTEMTIME {
  WORD wYear;
  WORD wMonth;
  WORD wDayOfWeek;
  WORD wDay;
  WORD wHour;
  WORD wMinute;
  WORD wSecond;
  WORD wMilliseconds;

} SYSTEMTIME,
*PSYSTEMTIME;
```

Miembros

wYear
Año (1601 - 30827).

wMonth
Mes.

Enero = 1
Febrero = 2
Marzo = 3
Abril = 4
Mayo = 5
Junio = 6
Julio = 7
Agosto = 8
Septiembre = 9
Octubre = 10
Noviembre = 11
Diciembre = 12

wDayOfWeek
Día de la semana.

Domingo = 0
Lunes = 1
Martes = 2
Miércoles = 3
Jueves = 4
Viernes = 5
Sábado = 6

wDay
Día del mes (1-31).

wHour
Hora (0-23).

wMinute
Minuto (0-59).

wSecond
Segundo (0-59).

wMilliseconds
Milisegundo (0-999).