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

Examen de Arquitectura de Computadores. (Telemática.) **Convocatoria de febrero, segunda parte: 03-02-2009**

Se dispone de un computador que utiliza un sistema de memoria virtual paginada con las siguientes características:

- Direcciones virtuales de 24 bits
- Direcciones físicas de 20 bits
- Tamaño de página de 512 posiciones (en cada posición de memoria se almacena un byte).

También se sabe que en dicho computador hay instalado un módulo de memoria que ocupa el 50% del espacio de direcciones físicas.

Teniendo en cuenta esta información, responde a las preguntas A, B y C:

— A) ¿Qué rango de direcciones físicas se corresponden con la memoria física? Contestar en hexadecimal.

`00000 – 7FFFF` **0,5**

— B) Un programa generado en este computador necesita dos páginas para albergar su código, otras dos para sus datos y una para su pila. El compilador siempre ubica la sección de código a partir de la dirección virtual 505000h, la de datos a partir de A00000h y la de pila a partir de 0071FFh. En esta situación, ¿cuál es la dirección virtual más significativa que puede ser accedida por el programa sin generar una excepción? Contestar en hexadecimal.

Nota: La pila crece hacia direcciones menos significativas.

`A003FF` **0,75**

— C) Se sabe que la tabla de páginas, además de guardar el marco de la memoria física donde se guarda cada página virtual, guarda 5 bits adicionales para dar soporte a los mecanismos de gestión y protección de memoria. ¿Cuánto ocupará (en KB) la tabla de páginas de cada proceso?

`64KB` **0,75**

Se dispone de otro computador que utiliza un sistema de memoria virtual paginada con las siguientes características:

- Direcciones virtuales de 16 bits
- Direcciones físicas de 12 bits
- Tamaño de página de 128 posiciones (en cada posición de memoria se almacena un byte).

Se sabe que el compilador siempre ubica la pila a partir de la última página virtual (la pila crece hacia direcciones virtuales menos significativas). En un determinado instante, el único proceso que hay en ejecución utiliza una página completa para el área de pila, y se sabe que ésta está en el marco 0A de memoria física.

— ¿Qué rango de direcciones virtuales ocupa la pila? Contestar en hexadecimal.

`FF80 - FFFF` **0,25**

— ¿Qué rango de direcciones físicas ocupa la pila? Contestar en hexadecimal.

`500 – 57F` **0,25**

Se dispone de otro computador más que utiliza un sistema de memoria virtual paginada con las siguientes características:

- Direcciones virtuales de 20 bits
- Direcciones físicas de 16 bits
- Tamaño de página de 512 posiciones (en cada posición de memoria se almacena un byte).

A continuación se muestra el estado en que se encuentra la tabla de páginas de un proceso en un momento dado. Las páginas virtuales que no se representan en la tabla **NO** son utilizadas por el proceso.

Tabla de Páginas

Nº Pag. Virtual	Presencia	Usuario / Supervisor	Read /Write	Nº Pag. Fis. / Offset Dis.
200	SI	Usuario	Read	1A
201	SI	Usuario	Read	1B
202	NO	Usuario	Read	Offset X
203	NO	Usuario	Read	Offset Y
50A	NO	Usuario	Read-Write	Offset Z
50B	SI	Usuario	Read-Write	07
50C	NO	Usuario	Read-Write	Offset N
7E0	SI	Supervisor	Read-Write	5C
7E1	SI	Supervisor	Read-Write	5D
7E2	SI	Supervisor	Read-Write	5E

A

— Indica a qué dirección o direcciones virtuales podría acceder el programa para realizar una escritura (cuando el sistema se encuentre en el estado descrito por la figura anterior) **sin generar una excepción**. Contesta NINGUNA si crees que se generaría una excepción en todos los casos.

A) A19DC

B) FC0CC

C) A1673

D) 74E73

C — **0,5**

— Define brevemente los siguientes conceptos

0,75

Fichero:

Es un conjunto de información relacionada, grabada en el sistema de almacenamiento secundario, a la que se hace referencia mediante un nombre.

Directorio:

Es un fichero especial que contiene información sobre otros ficheros y directorios.

Cilindro (de un disco duro):

Conjunto de todas las pistas que se ubican en la misma posición respecto al eje de giro del disco.

— Responde a las siguientes preguntas.

0,75

¿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).

¿Qué almacena la TLB?

Las entradas de la tabla de páginas más frecuentemente utilizadas.

— Responde a las siguientes preguntas.

0,5

¿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.

En el ámbito de los sistemas operativos, ¿qué se entiende por multitarea?

Que se permite la ejecución “concurrente” de múltiples procesos.

¿y por multiusuario?

Que se permite la interacción de varios usuarios simultáneamente con el sistema.

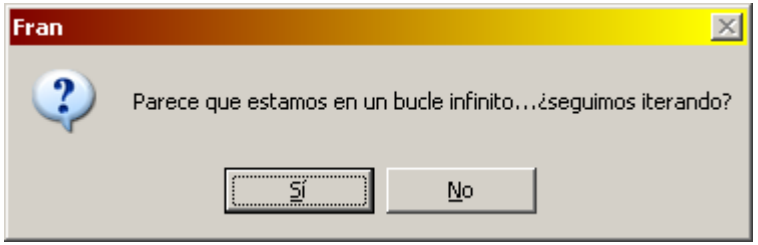
A **Apellidos** _____ **Nombre** _____ **DNI** _____

Examen de Arquitectura de Computadores. (Telemática.) **Convocatoria de febrero, segunda parte: 03-02-2009**

— Dibuja en el siguiente recuadro los estados por los que puede pasar un proceso durante su ciclo de ejecución y los eventos que pueden provocar que un proceso pase de uno de esos estados a otro.

0,75

A continuación se muestra un programa que entra irremediablemente en un bucle infinito. En cada una de las iteraciones, se pregunta al usuario si quiere seguir iterando o si prefiere dejar de sufrir y terminar el programa. Esta consulta se hace mediante una ventana como la mostrada a continuación.



Completa el código fuente del programa teniendo en cuenta las indicaciones expuestas en los comentarios.

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

main ()
{

      //Declaración de dos cadenas de caracteres para contener el
      //título y el texto de la ventana
      char titulo[50];

      char texto[]="Parece que estamos en un bucle infinito...
                  ¿seguimos iterando?";

      //Declaración de la variable utilizada para guardar el valor
       //retornado por la función utilizada para crear la ventana.
      //Indicar  en el recuardo el tipo de dato que le corresponde
      //a esta variable.
```

0,25 `int` resultado;

```
      //Pedir al usuario el nombre que aparecerá en el título
       //de la ventana
      printf("Escribe tu nombre: ");
```

0,5

```
scanf_s("%s",titulo,50);
```

```
      //Entrada en un bucle infinito
      //En cada iteración del bucle se debe crear una ventana
      //IDÉNTICA A LA MOSTRADA EN EL ENUNCIADO
      while (1){

            //Mostrar la ventana
```

1

```
resultado=MessageBox(NULL,
                     texto,
                     titulo,
                     MB_YESNO | MB_ICONQUESTION);
```

```
            //Si se ha pulsado “Sí” continuamos la ejecución
            //Si se ha pulsado “No”, terminar el programa
```

0,5

```
if (resultado==IDNO)
      ExitProcess(0);
```

```
      }
}
```

A

A continuación se muestra el listado de un programa que reserva espacio en memoria para almacenar un array de números enteros. La reserva de la memoria que albergará estos datos se realiza en tiempo de ejecución utilizando una llamada al sistema. Una vez reservada la memoria, se solicita al usuario que introduzca un número que se guardará en la posición 5 del vector (es decir, el sexto elemento, ya que el primer elemento está en la posición 0).

```
Introduce un numero: 40
El elemento 5 del vector es: 40
La memoria se libero correctamente
```

En el cuadro anterior se muestra un ejemplo de la salida por consola que debe generar este programa.

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

main ()
{
	int * p;

	//Escibir la sentencia que solicita al sistema operativo la
	//reserva de una página completa (4KB).
	//Dicha sentencia tiene que dejar el área de memoria lista
	//para poder escribir en ella
```

1

```
p=VirtualAlloc(
	NULL,
	4096,
	MEM_RESERVE | MEM_COMMIT,
	PAGE_READWRITE);
```

```
	//Si se produjo un fallo en la reserva, finalizamos el
	//programa

	if (p==NULL)
```

NO SE MUESTRA ESTA PARTE

```
	//Se pide al usuario que escriba un número que será
	//almacenado en la posición 5 del vector
	printf("Introduce un numero: ");

	//Escribir las instrucciones necesarias para que se lea
	//por teclado un número y se almacene en la posición 5 del
	//vector. A continuación se muestra por pantalla para
	//comprobar que la asignación se realizó correctamente.
	//La salida producida DEBE coincidir con la mostrada
	//en el enunciado
```

1

```
scanf_s("%d",p+5);
printf("El elemento 5 del vector es: %d\n", *(p+5));
```

```
	if(VirtualFree(p,0,MEM_RELEASE))
		printf("La memoria se libero correctamente\n");
	else
		printf("Se ha producido un error en la liberacion\n");

}
```