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

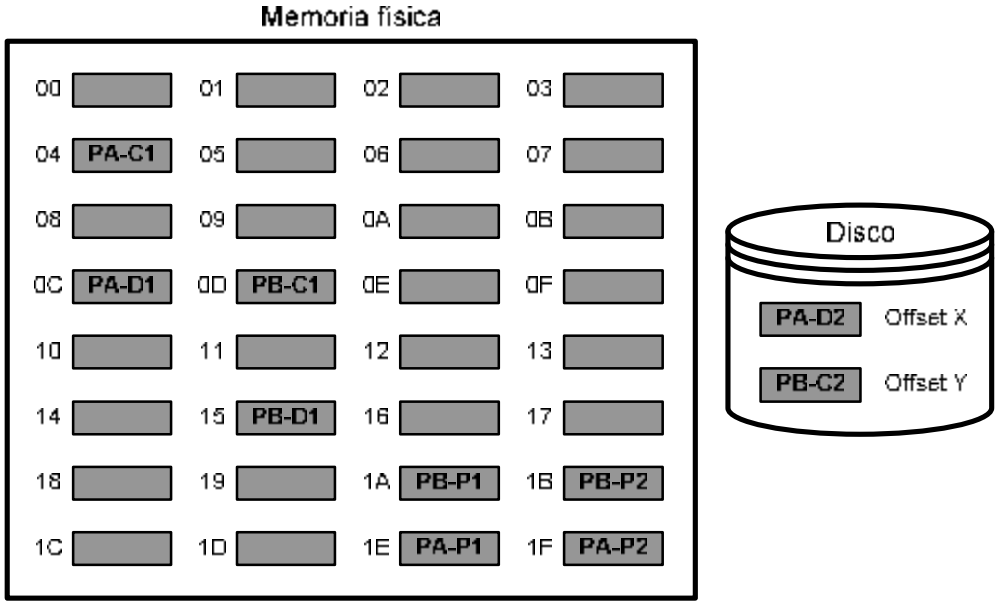
Examen de Arquitectura de Computadores. (Telemática.) **Convocatoria de febrero, segunda parte: 09-02-2008**

La MMU de un computador utiliza un sistema de traslación de direcciones con las siguientes características:

Dirección virtual: 20 bits — Dirección física: 16 bits — Tamaño de página: 512 bytes

En la figura que se proporciona a continuación se muestra la organización de la memoria física del sistema. Los rectángulos grises representan los marcos de página. A la izquierda de cada marco se indica el número que le corresponde.

En el instante representado en la figura, se encuentran en ejecución en el sistema dos procesos, denominados PA y PB. El proceso PA utiliza 5 páginas, denominadas PA-C1, PA-D1, PA-D2, PA-P1 y PA-P2. Las páginas se nombran utilizando como prefijo el nombre del proceso (PA en este caso), seguido de una letra (C, D o P), que indica si la página es de código, datos o pila, y finalmente un número indicando el orden que ocupa la página en la sección del programa a la que pertenece. Así por ejemplo, las páginas PA-D1 y PA-D2 indican las páginas de datos del proceso PA, encontrándose ubicadas en la sección de datos primero PA-D1 y después, PA-D2. El proceso PB utiliza otras 5 páginas que se nombran utilizando las mismas reglas que las usadas para el proceso PA. La figura muestra la ubicación de las páginas de PA y PB, tanto en la memoria física como en el disco.



Se sabe que en este sistema las diferentes secciones de los programas se direccionan de forma estándar comenzando en las siguientes direcciones:

Código: 10000 — Datos: 20000 — Pila: 08000

Además, debe tenerse en cuenta que las páginas de código y datos se asignan en el orden de las direcciones crecientes, pero la páginas de pila se asignan en el orden inverso, es decir, en el de las direcciones decrecientes.

Finalmente debe indicarse que el sistema se encuentra configurado para que cuando se produzca un fallo de página, la página que se mueve del disco a la memoria física se ubique en el marco de página más significativo que se encuentre libre en ese momento.

Teniendo en cuenta toda esta información contesta a las preguntas A, B y C.

— A) Rellena las tablas de páginas de los procesos PA y PB mostradas a continuación. Observa que en cada tabla sólo se representa 5 entradas: éstas corresponden a las 5 páginas utilizadas por cada proceso. Debes rellenar los tres campos de cada entrada, incluido en campo Nº de página virtual. Ten en cuenta que aunque todas las entradas de las tablas se representan separadas, algunas de ellas pueden ser consecutivas.

1

Tabla de Páginas de PA

Nº Pag. Virtual	Pre-sencia	Nº Pag. Fis. / Offset Dis.
03F	Si	1F
040	Si	1E
080	Si	04
100	Si	0C
101	No	Offset X

Tabla de Páginas de PB

Nº Pag. Virtual	Pre-sencia	Nº Pag. Fis. / Offset Dis.
03F	Si	1B
040	Si	1A
080	Si	0D
081	No	Offset Y
100	Si	15

— B) Determina el porcentaje del espacio de direcciones físico ocupado por la memoria física de este sistema.

0,5

25%

— C) Se sabe que las dos páginas de datos del proceso PA contienen un vector de 256 enteros. Cada entero se almacena mediante 4 bytes. Indica las direcciones FÍSICAS generadas por la MMU al accederse a los elementos del vector que se indican a continuación. Contesta en hexadecimal.

0,75

Diercción fis. correspondiente a vector[5]: 1814

Dirección fis. correspondiente a vector[129]: 3A04

A

— Indica los diferentes estados por los que puede pasar un proceso durante su ejecución, y define brevemente cada uno de ellos.

0,75

Nuevo: El proceso se acaba de crear, pero aún no ha sido admitido en el grupo de procesos ejecutables por el sistema operativo.

Listo: El proceso está esperando ser asignado al procesador para su ejecución.

En ejecución: El proceso tiene la CPU y ésta ejecuta sus intrucciones.

En espera: El proceso está esperando a que ocurra algún sucero, como por ejemplo la terminación de una operación de E/S.

Terminado: El proceso ha sido sacado del grupo de procesos ejecutables por el sistema operativo.

— Contesta las siguientes preguntas breves:

0,75

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

Copiar la página a la que se hacho 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 tuiliza?

Indica si la página corresondiente es de solo lectura o de lectura/escritura.

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

¿Qué almacena la TLB?

Las entradas de las tablas de páginas más frecuentemente utiliadas.

— Define los tres objetivos básicos de un sistema operativo.

0,75

Proporcionar una interfaz amigable para la interacción entre el usuario y el computador.

Proporcionar un entorno de funcionamiento para los programas. Así el sistema operativo proporciona un conjunto de servicios que pueden ser solicitados por los programas.

Coordinar el uso de los recursos hardware del computador entre los programas que se encuentran en ejecución en cada instante.

— Contesta las siguientes preguntas breves:

0,5

Dirección más significativa que puede ser utilizada por un proceso Windows:
7FFE FFFF

Nombre que reciben los programas que hacen sus operaciones de E/S en una consola Win32:
CUI

Escribe una sentencia que inicialice un puntero llamado p de tipo int con la dirección 00010000h. La sentencia debe compilar sin generar ningún tipo de aviso.

```
p=(int *)0x00010000;
```

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

Examen de Arquitectura de Computadores. (Telemática.) **Convocatoria de febrero, segunda parte: 09-02-2008**

El objetivo de un programa es poner en ejecución un ejecutable, cuyo nombre es introducido por el usuario a través de la consola. Para ello, el programa utiliza una función llamada *EjectuaEjecutable()*. Esta función recibe tres parámetros: la dirección de un buffer conteniendo el nombre del ejecutable a ejecutar y las dimensiones X e Y de la consola que se asociará a dicho ejecutable. La función *main()* del programa simplemente pide al usuario que introduzca el nombre del ejecutable y las dimensiones X e Y de la consola y llama a *EjecutaEjecutable()* con los parámetros apropiados.

A continuación se indica la información mostrada en la consola por el programa cuando el usuario solicita ejecutar el ejecutable *prueba1.exe*, con una consola de 700 x 700, y generándose un PID igual a 2176.

```
Nombre ejecutable: prueba1.exe
Dimension_X: 700
Dimension_Y: 700
PID = 2176
```

A continuación se proporciona el listado del programa. Debes rellenar los huecos del programa teniendo en cuenta la información enviada por éste a la consola (indicada en el recuadro anterior), así como los comentarios indicados en el listado.

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

void EjecutaEjecutable(char *, int, int);

main()
{

    char nombreEjecutable[40];
    int  dimension_X=0, dimension_Y=0;

    // Solicitar el nombre del ejecutable a ejecutar
    // y las dimensiones de su consola
```

0,5

```
    printf("Nombre ejecutable: ");
    scanf_s("%s", nombreEjecutable, 40);
    printf("Dimension_X: ");
    scanf_s("%d", &dimension_X);
    printf("Dimension_Y: ");
    scanf_s("%d", &dimension_Y);
```

```
    EjecutaEjecutable( nombreEjecutable, dimension_X, dimension_Y );
    _getch();
}

void EjecutaEjecutable(char *nombreEje, int dimX, int dimY)
{
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    int ExitoApertura;

    // Inicializacion de la estructura STARTUPINFO

    // En este hueco debes rellenar los campos de STARTUPINFO
    // necesarios para que al ejecutable se le asigne una consola
    // con las dimensiones indicadas por el usuario. La
    // inicialización del resto de los campos de STARTUPINFO no
    // se muestra en este código.
```

0,75

```
    si.dwXSize =            dimX;
    si.dwYSize =            dimY;
    si.dwFlags =            STARTF_USESIZE;
```

```
    ExitoApertura = CreateProcess(
                        NULL,               // pszApplicationName
                        nombreEje,          // pszCommandLine
                        NULL,               // psaProcess
                        NULL,               // psaThread
                        TRUE,               // bInheritHandles
                        CREATE_NEW_CONSOLE, // fdwCreate
                        NULL,               // pvEnvironment
                        NULL,               // pszCurDir
                        &si,                // psiStartInfo
                        &pi      );         // ppiProcInfo

    // Si ha habido éxito en la paertura se imprime el PID del
    // proceso generado. En el caso contrario, se envía un mensaje
    // indicando que ha habido un fallo en la apertura
```

0,75

```
    if ( ExitoApertura )
    {
        printf("PID = %d\n\n", pi.dwProcessId);
    }
    else
        printf("Fallo en la apertura del programa\n\n");
```

```
}
```

A

El objetivo de un programa es procesar una serie de estructuras recibidas a través de la consola. Cada una de estas estructuras almacena tres datos relativos a una persona. Estos datos son el nombre, la nacionalidad y la edad. El programa, primero, recibe a través de la consola el número de personas a procesar. Entonces reserva y compromete una región de memoria virtual para almacenar el número de personas indicado. La región comprometida debe tener el tamaño suficiente para almacenar el número de personas indicado, pero no debe ser mayor de lo estrictamente necesario. Después el programa ejecuta las instrucciones necesarias para que se produzca la entrada de las estructuras de tipo persona, que se almacenan en la región de memoria comprometida. A continuación, el programa pide al usuario que introduzca el índice de una de las personas introducidas y el programa muestra el nombre de la persona correspondiente. Finalmente se libera la región de memoria comprometida.

A continuación se muestra la información enviada por el programa a la consola cuando se le pide al programa que procese una sola persona, cuyos datos son los siguientes: nombre = pepe, nacionalidad = español, y edad = 30.

```
Numero de personas a procesar: 1

========== Introducir personas ==========
Persona[0]
nombre: pepe
nacionalidad: español
edad: 30

Introduce un indice de persona: 0
Persona[0] = pepe

Liberacion OK
```

El listado del programa perfilado mediante comentarios se muestra a continuación. Debes rellanar los huecos existentes en el programa, poniendo atención a los comentarios del mismo. La información enviada a la consola por el programa debe ser coherente con la representada en el recuadro anterior.

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

typedef struct _PERSONA
{
    char nombre[100];
    char nacionalidad[24];
      int  edad;
} PERSONA;
```

```
main()
{
    PERSONA *p, *q;      // Para apuntar a la region comprometida
    int  num_personas;   // Número de personas a procesar
    int  ind_persona;    // Índice de persona
    int  i;              // Contador

    // Pedir por pantalla el numero de personas a procesar
    printf("\n\nNumero de personas a procesar: ");
    scanf_s("%d", &num_personas);

    // Reservar y comprometer la memoria necesaria para almacenar
    // las pesonas a procesar. Hacer que el SO elija la region
```

0,5

```
    p=VirtualAlloc(NULL,
                   num_personas*128,
                   MEM_RESERVE | MEM_COMMIT,
                   PAGE_READWRITE);
    q=p;

    // Código para introducir personas
    // p no se modifica
    // q se modifica de la forma apropiada para acceder a los
    // miembros de las sucesivas personas
```

1

```
    printf("\n========== Introducir personas ==========\n");
    for(i=0; i<num_personas; i++)
    {
        printf("Persona[%d]\n", i);
        printf("nombre: ");
        scanf_s("%s", q->nombre, 100);
        printf("nacionalidad: ");
        scanf_s("%s", q->nacionalidad, 24);
        printf("edad: ");
        scanf_s("%d", &(q->edad));
        printf("\n");
        q++;
    }

    // Pedir por pantalla un índice de persona
    printf("\nIntroduce un indice de persona: ");
    scanf_s("%d", &ind_persona);
```

```
    // Código para mostrar el nombre de la persona
    // correspondiente al índice introducido
    // Debe utilizarse obligatoriamente el puntero p

    if ( ind_persona < num_personas )
    {
        printf("Persona[%d] = %s\n", ind_persona,
                                     (p+ind_persona)->nombre);

    }
    else
        printf("Error, el indice es demasiado grande\n");

    // liberar completamente la memoria indicando si la operación
    // tiene éxito o no
    if ( VirtualFree(p, 0, MEM_RELEASE) )
       printf("\nLiberacion OK\n");
    else
       printf("\nFallo en la liberacion de memoria\n");
}
```

0,5

— En una determinada ejecución del programa anterior, la variable *num_personas* tomó el valor 130 (decimal). Además se sabe que la última estructura de tipo persona introducida durante esta ejecución (la de índice 129) se ubicó en la página virtual 005A4. A partir de estos datos, determina el valor retornado por *VirtualAlloc()* en esta ejecución del programa. Contesta en hexadecimal.

0,5

005A0000

A continuación se muestra el listado de un programa simple cuyas variables globales se ubican todas contiguas en la sección de datos a partir de la dirección 00417000.

```
#include <stdio.h>

int A=0, B=5, C=10, *p;

main()
{


    p=&B;

    printf("%p", &p);  // Primer printf
    printf("%p",  p);  // Segundo printf
    printf("%p", *p);  // Tercer printf
}
```

— Determina el valor que será impreso en la consola por cada una de las sentencias *printf()* del programa.

0,5

Primer printf: **0041700C**

Segundo printf: **00417004**

Tercer printf: **00000005**