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

Examen de Arquitectura de Computadores. (Telemática.) **Convocatoria de septiembre: 11-09-2007**

```
.386
.MODEL FLAT, stdcall
ExitProcess PROTO, :DWORD

.DATA
matrizA  DB  3,  1, -9
         DB  14, 2, -7
         DB  9,  7, 6

matrizB  DB  1,  4, 21, 7
         DB  20, 3, 2,  6
         DB  -1, 7, 2,  -9
         DB  15, 8, 5,  -3

mayorSumaDiagonal  DD  0

.CODE
sumaDiagonal PROC
  push  ebp
  mov   ebp, esp
  sub   esp, 4           ; crear espacio para la variable local
                         ; para contener el orden de la matriz
  push  esi
  push  ecx
  push  ebx

  ; Recuperar parámetros
  mov   esi, [ebp+12]    ; ESI apunta a al comienzo de la matriz
  mov   ecx, [ebp+8]     ; ECX se iniciliaza con el orden de la
                         ; matriz

  xor   ebx, ebx         ; EBX auxiliar para direccionar la matriz
  xor   eax, eax         ; EAX para retornar la suma de la diagonal

  ; inicilizar la variable local con el orden de la matriz
```

No se muestra esta instrucción

```
bucle:
  add al, [esi+ebx]
  ; actualizar EBX de la forma apropiada haciendo uso
  ; de la variable local. ESI permanece fijo
  (--1--)

  loop bucle

  ; Restauración de registros y retorno
  pop ebx
  pop ecx
  pop esi  ◄◄◄

  (--2--)

sumaDiagonal ENDP

inicio:
```

Programa principal

```
END inicio
```

El programa anterior calcula la suma de las diagonales de las dos matrices definidas en su sección de datos y almacena la suma mayor en la variable *mayorSumaDiagonal*.

Para llevar a cabo el procesamiento indicado, el programa utiliza el procedimiento *sumaDiagonal*. Este procedimiento recibe dos parámetros a través de la pila. Estos parámetros, que se indican a continuación según su orden de apilación, son los siguientes:

1) La dirección de comienzo de la matriz a procesar.
2) El orden de la matriz (se trata del número de filas o de columnas de la matriz. Así por ejemplo, la matriz A del programa es de orden 3.)

Las diagonales de las matrices están formadas por sus elementos (0, 0), (1, 1), (2, 2) y así sucesivamente, donde (0, 0) significa (fila 0, columna 0). Así la diagonal de la matriz A está formada por los números 3, 2 y 6.

El procedimiento devuelve como resultado la suma de los miembros de la diagonal de la matriz cuya dirección y orden se le pasan como parámetros. El resultado es devuelto en el registro EAX.

Para llevar a cabo su procesamiento, el procedimiento utiliza una variable local de 32 bits. Dicha variable se actualiza al comienzo del procedimiento con el orden de la matriz. Esta variable se usa para modificar de la forma apropiada el registro EBX, que se utiliza en el acceso a los sucesivos elementos de la diagonal.

El procedimiento está diseñado siguiendo el esquema estandarizado de construcción y destrucción del marco de pila. Además, el propio procedimiento es el encargado de destruir los parámetros.

Como datos adicionales se conoce que la dirección de comienzo de la sección de datos del programa es 00404000 y que el valor del registro SP, justo cuando el programa comienza a ejecutarse, es 0012FFC4.

Teniendo en cuenta toda esta información contesta a las pregustas que se proporcionan a continuación.

A

— Escribe el programa principal completo necesario para el programa. Para ello, ten en cuenta los comentarios que se indican. Para obtener la máxima puntuación en esta pregunta debes utilizar en tu código la menor cantidad de registros posible.

1

```
inicio:
  ; Obtener la suma de la diagonal de matrizA
  ; y guardar en EBX esta suma
  push  OFFSET matrizA
  push  3
  call  sumaDiagonal
  mov   ebx, eax


  ; Obtener la suma de la diagonal de matrizB
  push  OFFSET matrizB
  push  4  ; dimesión de la matriz
  call  sumaDiagonal


  ; Actualiazar mayorSumaDiagonal
  ; con el valor apropiado
  cmp   eax, ebx
  jg    sigue1
  mov   [mayorSumaDiagonal], ebx
  jmp   sigue2
sigue1:
  mov   [mayorSumaDiagonal], eax
sigue2:

  ; Retorno al Sistema Operativo
  push 0
  call ExitProcess
END inicio
```

— ¿Qué instrucción o instrucciones son necesarias en el hueco (—**1**—) del listado?

0,5

```
add ebx, [ebp-4]
inc ebx
```

— ¿Qué instrucciones son necesarias en el hueco (—**2**—) del listado?

0,5

```
mov esp, ebp
pop ebp
ret  8
```

— Determina el valor que tendrá el puntero de pila justo después de la ejecución de la instrucción "**`pop esi`**", marcada en el listado con el símbolo ◄◄◄. Contesta en hexadecimal.

0,5

```
0012ffB0
```

— Imagina que las matrices de este programa hubieran sido definidas utilizando datos de tipo doble palabra, en vez de datos de tipo byte. Teniendo en cuenta este supuesto, determina el byte que se almacenaría en la dirección 0040400C de la sección de datos. Contesta en hexadecimal.

0,5

```
0E
```

A continuación se muestra el listado de un programa C muy simple.

```
int A=2, B=0, *p=&A;

main()
{
    B = *p; ♥♥♥
}
```

— Teniendo en cuanta el programa anterior, escribe el conjunto de instrucciones ensamblador que generaría el compilador de C al compilar la sentencia marcada con el símbolo **♥♥♥**.

0,5

```
mov  eax, [p]
mov  ebx, [eax]
mov  [B], ebx
```

— Codifica la instrucción "**`mov ecx, [ebx-20]`**".

0,5

```
8B 4B EC
```

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

Examen de Arquitectura de Computadores. (Telemática.) **Convocatoria de septiembre: 11-09-2007**

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

Dirección física: 16 bits

Dirección virtual: 16 bits

Tamaño de página: 1 KB

Además se sabe que la memoria física ocupa el 25% del espacio de direcciones físico y se mepea a partir de la dirección 0000 de éste.

En un momento dado se encuentra en ejecución en este sistema un proceso cuya tabla de páginas es la que se muestra a la derecha.

Tabla de Páginas

Nº Pag. Virtual	Pre-sencia	Nº Pag. Fis. / Offset Dis.	
00	No	---	
•	•	•	
10	Si	01	P1
•	•	•	
18	Si	00	C1
•	•	•	
20	Si	03	D1
21	Si	04	D2
22	No	Offset X	D3
23	No	Offset Y	D4
•	•	•	
3F	No	---	

El proceso utiliza una página de código (C1), una página de pila (P1) y cuatro páginas de datos (D1, D2, D3 y D4). A la derecha de las entradas de la tabla de páginas se indica qué páginas virtuales son utilizadas por el proceso para direccionar sus páginas de código, datos y pila. (Así por ejemplo, según se observa en la figura anterior, la página de pila P1 es direccionada mediante la página virtual 10). Todas las entradas de la tabla de páginas no mostradas corresponden a páginas no utilizadas por el proceso.

El sistema está configurado para que al producirse un fallo de página, la página que se mueve del disco hacia la memoria física se ubique en el marco de página física menos significativo que se encuentre libre.

Finalmente, debe considerarse que los únicos marcos de página física ocupados son aquellos utilizados por el proceso.

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

— A) Determina la dirección virtual más significativa que podría generar el proceso al acceder a un dato de su sección de datos.

8FFF **0,5**

— B) Determina el rango de direcciones del último marco de página de la memoria física. (Ejemplo de respuesta 1FB1 – 1FCC.)

3C00 – 3FFF **0,5**

— C) Si la MMU recibe la dirección virtual 8C33, determina o bien la dirección física generada, o si crees que se produce una violación de acceso a memoria, contesta VIOLACIÓN DE ACCESO.

0833 **0,5**

— D) Imagina que en este sistema estuviera disponible una función *VirtualAlloc()* similar a la de la plataforma Windows, salvo en los siguientes aspectos:

- El campo en el que se indica la dirección, que sería de 16 bits, en vez de 32 bits.
- La granularidad de reserva, que sería de 1KB (una página) en vez de 64KB.

Si el proceso ejecutase:

*VirtualAlloc((void *)0xC000, 4000, MEM_RESERVE, PAGE_READWRITE);*

Determina el rango de páginas virtuales reservadas por la función. El rango debe indicarse mediante números de página, es decir, es un rango de páginas, no de direcciones. (Ejemplo de respuesta: 20 – 26.)

30 - 33 **0,5**

A

— Escribe una función *main()* que imprima en pantalla la fecha actual según el formato día–mes–año. Así por ejemplo, si la fecha actual es el 11 de septiembre de 2007, el programa debe escribir en pantalla lo siguiente:

```
Fecha: 11-9-2007
```

1

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

main()
{

  SYSTEMTIME fecha;

  GetLocalTime(&fecha);

  printf("Fecha: %d-%d-%d\n", fecha.wDay, fecha.wMonth, fecha.wYear);



}
```

A continuación se proporciona el listado de un programa simple escrito en lenguaje C.

```
#include <stdio.h>

int vector[5] = {0, 2, 4, 6, 8};

main()
{
       int *p, i;

       p=vector;

       printf("%p", p+3); ◄◄◄

       // Sumar 2 a todos los elementos del array vector

       (--1--)
```

Se conocen las siguientes direcciones de elementos de este programa:

Dirección del primer elemento de vector = 00417000.

Dirección de la variable correspondiente al puntero p = 0012FF60.

A partir de toda esta información contesta a las preguntas A) y B).

— A) ¿Qué valor imprimirá en pantalla la sentencia *printf()* del programa, marcada con el símbolo ◄◄◄?

0,5

```
0041700C
```

— B) Escribe la sentencia necesaria para rellenar el hueco (—**1**—). Se trata de un bucle *for*, que utiliza como índice la variable *i* definida previamente. Dentro del bucle NO puedes utilizar la etiqueta *vector*. Tampoco puedes utilizar los símbolos '[', ni ']'. (Se trata de que utilices una sintaxis de tipo puntero.)

0,5

```
for(i=0; i<=4; i++)
   *(p+i) += 2;
```

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

Examen de Arquitectura de Computadores. (Telemática.) **Convocatoria de septiembre: 11-09-2007**

— Indica y describe brevemente los componentes básicos de un disco duro

0,5

Platos: Están formados por una aleación rígida de aluminio y recubiertos por una capa de material magnético sobre la que se graba la información.

Motor de giro: Su objetivo es hacer girar los platos a velocidad constante.

Cabezas de lectura/escritura: Su objetivo es escribir información sobre la superficie de los platos y leer de ellos. Hay una cabeza por cada superficie.

Brazo: Su objetivo es servir de soporte para la cabeza.

Actuador: Es un servomotor encargado de mover los brazos para posicionar las cabezas en las posiciones de los platos requeridas.

— Define la TLB indicando claramente su objetivo

0,5

Es una memoria de alta velocidad totalmente integrada con la MMU, que almacena las entradas de la tabla de páginas más frecuentemente utilizadas. Así las direcciones virtuales pueden ser traducidas sin necesidad de acceder a la memoria física. Un mecanismo hardware permite actualizar dinámicamente el TLB con las entradas de la tabla de páginas más frecuentemente utilizadas.

— Contesta a las preguntas que se indican en el siguiente cuadro:

0,5

Con relación a los tipos de programas Win32, indica el significado del acrónimo CUI:

Console User Interface

Indica el nombre del programa que es la interfaz gráfica de comandos en la plataforma Windows:

explorer.exe

Nombre de los campos o miembros del tipo de estructura STARTUPINFO que se utilizan para determinar el tamaño de la consola que se asocia a un proceso puesto en ejecución por CreateProcess():

dwXSize, dwYSize

— Contesta a las siguientes preguntas sobre las interrupciones NO enmascarables:

0,5

¿Para qué se utilizan?

Son utilizadas para indicar al procesador la ocurrencia de errores hardware graves, y cuando se producen el procesador las acepta siempre.

Nombre de la línea que tienen asociada:

NMI

Número de interrupción que tienen asociado:

2