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

Examen de Arquitectura de Computadores. (Telemática.) **Convocatoria de febrero, primera parte: 13-02-2007**

A continuación se muestra el listado de un programa cuyo objetivo es sumar dos números decimales de cuatro dígitos, almacenados en la sección de datos del programa. Además, debe tenerse en cuenta que estos números no se encuentran almacenados como números en sí mismo, sino como cadenas de cuatro caracteres. Los caracteres de estas cadenas sólo pueden ser dígitos entre el 0 y el 9, ambos inclusive, por lo que estas cadenas representan números decimales.

Para llevar acabo la suma de los dos números expresados mediante cadenas, previamente debe convertirse cada cadena al número que representa. Para ello se define un procedimiento llamado *Convierte*. Este procedimiento recibe dos parámetros a través de la pila. Estos parámetros se indican a continuación según su orden de apilación: 1) la dirección de la cadena a procesar y 2) la dirección de un *array* conteniendo las cuatro primeras potencias de 10 que serán usadas en el procedimiento de conversión. Para entender cómo se lleva a cabo la conversión nos fijaremos en la cadena *CadenaNumero1* de la sección de datos del programa. Se trata de "2124". Para procesar esta cadena, el procedimiento *Convierte* toma en primer lugar el carácter '2', lo convierte en el número 2 y luego lo multiplica por el primer valor del *array* de potencias, que es 1000. El resultado obtenido lo almacena en un acumulador. Después toma el siguiente carácter, '1', lo convierte en número y lo multiplica por 100, agregando este resultado al acumulador. Este procedimiento se repite hasta que se han procesado los cuatro caracteres de la cadena, obteniéndose finalmente el resultado. El procedimiento *Convierte* retorna el resultado en el registro EAX.

Para llevar a cabo las multiplicaciones necesarias, el procedimiento *Convierte* utiliza la instrucción "**mul {R32|M32}**". Esta instrucción multiplica el contenido del registro EAX por el contenido del operando R32 (registro de 32 bits) o M32 (posición de memoria de 32 bits) indicado en la instrucción. El resultado (de 64 bits) se almacena en EDX:EAX (los 32 bits más significativos en EDX y los 32 menos significativos en EAX). En este programa, como los números a multiplicar son pequeños, EDX siempre tendrá 0 después de ejecutar la instrucción **mul**, por lo que no se hará nada con este registro después de **mul**.

Como datos adicionales se sabe que la sección de datos se ubica a partir de la dirección 00404000 y que justo antes de que el programa comience a ejecutarse ESP=0012FFC4.

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

.DATA
  Resultado        dd 0
  Potencias        dd 1000, 100, 10, 1
  CadenaNumero1    db "2124"
  CadenaNumero2    db "0672"
```

```
.CODE
Convierte PROC
  push ebp
  mov  ebp, esp
  push esi
  push edi
  push ebx
  push ecx
  push edx

  mov  esi, [ebp+12]
  mov  edi, [ebp+8]
  xor  ebx, ebx ; Acumulador
  mov  ecx, 4

bucle:
  xor  eax, eax
  mov  al, [esi]
  ; Se convierte el ascii del numero en el numero
  sub  al, '0'
  ; Se multiplica el digito por la potencia que le corresponde
  mul  DWORD PTR [edi]

  ; Se acumula el valor correspondiente al dígito procesado
  add  ebx, eax

  inc  esi
  add  edi, 4
  loop bucle

  ; Devolver el resultado
  mov  eax, ebx    ◆◆◆

  pop  edx
  pop  ecx
  pop  ebx
  pop  edi
  pop  esi
  pop  ebp
  ret
Convierte ENDP

Inicio:
```

PROGRAMA PRINCIPAL

```
END Inicio
```

A

— Escribe el programa principal completo necesario para el programa. Para obtener la máxima puntuación en esta pregunta debes utilizar en tu código la menor cantidad de registros posible.

2

```
; Convertir CadenaNumero1
push OFFSET CadenaNumero1
push OFFSET Potencias
call Convierte
add  esp, 8
mov  [resultado], eax


; Convertir CadenaNumero2
push OFFSET CadenaNumero2
push OFFSET Potencias
call Convierte
add  esp, 8
add  [resultado], eax


; Retorno al sistema operativo
push    0
call    ExitProcess
```

— Determina cuál es el valor más grande que alcanza el registro ESI durante la ejecución de este programa (se entiende que este registro no se utiliza en la parte del programa correspondiente al programa principal.)

1

```
0040401C
```

— Determina el contenido de las posiciones de memoria que se indican a continuación mientras se está ejecutando el procedimiento *convierte* para procesar *CadenaNumero1*. Contesta con 2 dígitos hexadecimales para cada una de ellas.

1

```
0012FFBE: 40
0012FFBF: 00
0012FFC0: 14
0012FFC1: 40
```

— ¿Qué valor tiene el registro EBP cuando se ejecuta la instrucción "**mov eax, ebx**", marcada en el listado del programa con el símbolo ◆◆◆? Contesta en hexadecimal.

1

```
0012FFB4
```

— Codifica la instrucción "**mov [ebx+esi*2+2], DWORD PTR 4**".

1

```
C7 44 73 02 04 00 00 00
```

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

```
// Variables globales
int vector[3] = {0, 0, 0};
int A=3, B=4;

main()
{
    int *p;

    vector[1] = A+B; ♥♥♥

    p = vector;

    *p = B+0x1B; ♦♦♦
}
```

Teniendo en cuenta el programa anterior, contesta a las siguientes preguntas.

— Escribe el conjunto de instrucciones ensamblador que generaría el compilador de C al compilar la sentencia "vector[1] = A+B", marcada con el símbolo ♥♥♥ en el listado.

1

```
mov  eax, [A]
add  eax, [B]
mov  [vector+4], eax
```

— Escribe el conjunto de instrucciones ensamblador que generaría el compilador de C al compilar la sentencia "*p = B+0x1B", marcada con el símbolo ♦♦♦ en el listado.

1

```
mov  eax, [B]
add  eax, 1Bh
mov  ecx, [ebp-4]
mov  [ecx], eax
```

A Apellidos _____ Nombre _____ DNI _____

Examen de Arquitectura de Computadores. (Telemática.) **Convocatoria de febrero, primera parte: 13-02-2007**

— Contesta a las siguientes preguntas sobre las excepciones de tipo fallo en la arquitectura IA-32

1

Objetivo:

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

Tratamiento de los fallos recuperables:

El SO subsana el fallo y retorna a la instrucción que lo provocó, que se ejecutará sin problema.

Ejemplo de excepción que provoca un fallo recuperable:

Fallo de página

Tratamiento de los fallos NO recuperables:

La insturcción que provocó el fallo queda señalizada (útil a efectos de depuración). El SO termina el programa donde se produjo el fallo y planifica otro programa diferente.

Ejemplo de excepción que provoca un fallo NO recuperable:

Violación de protección o fallo de protección general

— Indica los nombres de las excepciones que se asignan a las siguientes entradas de la IDT:

0,5

Entrada 0: División por 0

Entrada 1: Excepción modo traza

Entrada 6: Código de operación inválido

Entrada 8: Doble falta

— 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.