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

Examen de Arquitectura de Computadores. (Telemática.) **Parcial de diciembre: 12-12-2007**

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

.DATA
cadenaOrigen   DB  "1FccX3091:!A71:", 0
cadenaDestino  DB  16  DUP(0)
numDigitos     DD  0

.CODE
inicio:
  ; Llamada a copiaDigitos
  push OFFSET cadenaOrigen
  push OFFSET cadenaDestino
  call copiaDigitos
  (--1--)

  ; Actualizar numDigitos
  mov  [numDigitos], eax   ♥♥♥

  ; Retorno al Sistema Operativo
  push 0
  call ExitProcess

copiaDigitos PROC
  push ebp
  mov  ebp, esp
  push esi
  push edi
  push edx
  push ebx

  ; Cargar parámetros en EDI y ESI
  mov  edi, [ebp+8]
  mov  esi, [ebp+12]   ♠♠♠

  ; Resetear contador de dígitos
  xor  edx, edx
  ; Resetear auxiliar
  xor  ebx, ebx
bucle:
  cmp  [esi], BYTE PTR 0
  je   SHORT  fuera
  mov  bl, [esi]
  push ebx
  call esDigito
  cmp  eax, 0
  je   SHORT  noEsDigito
  ; Sí es dígito
  (--2--)

noEsDigito:
  inc  esi
  jmp  bucle
fuera:

  ; Poner terminador en la cadena destino
  mov  [edi], BYTE PTR 0
  ; Retornar número de dígitos
  mov  eax, edx

  ; Finalizar procedimiento
  pop  ebx
  pop  edx
  pop  edi
  pop  esi
  pop  ebp
  ret
copiaDigitos ENDP

esDigito PROC
  push ebp
  mov  ebp, esp
  push edx

  mov  edx, [ebp+8]

  cmp  dl, '0'
  jb   SHORT _noEsDigito
  cmp  dl, '9'
  ja   SHORT _noEsDigito
  ; Es dígito
  (--3--)

sigue:

  ; Finalizar procedimiento
  pop  edx
  pop  ebp
  ret  4
esDigito ENDP

END inicio
```

A

El listado anterior corresponde a un programa cuyo objetivo es procesar una cadena de caracteres definida en la sección de datos, obteniendo de ella aquellos caracteres que son dígitos numéricos y copiándolos en otra cadena, también de la sección de datos. Los dígitos numéricos son aquellos caracteres comprendidos entre el '0' (ASCII 30h) y el '9' (ASCII 39h), ambos inclusive. Las cadenas de caracteres se encuentran terminadas con el número 0.

Para llevar a cabo el procesamiento requerido, el programa utiliza el procedimiento *copiaDigitos*. Este procedimiento recibe dos parámetros a través de la pila: 1) la dirección de la cadena que se quiere procesar (cadena origen), y 2) la dirección de la cadena en la que se desea almacenar los dígitos obtenidos (cadena destino).

El procedimiento *copiaDigitos* va procesando uno a uno los caracteres de la cadena origen. Si el carácter procesado es un dígito, lo copia en la cadena destino e incrementa un contador que indica el número de dígitos copiados. Si el carácter procesado no es un dígito, no se hace nada con él. Para determinar si el carácter procesado es dígito, el procedimiento *copiaDigitos* utiliza otro procedimiento, denominado *esDigito* que se describe en el párrafo siguiente. El procedimiento *esDigito* se ejecuta de forma anidada en *copiaDigitos*. *copiaDigitos* devuelve como resultado en el registro EAX el número de dígitos copiados.

El procedimiento *esDigito* recibe un único parámetro a través de la pila: el carácter a procesar. El procedimiento determina si dicho carácter se encuentra en el intervalo correspondiente a los dígitos numéricos. Si esto es así, devuelve en el registro EAX un 1. En el caso contrario, devuelve en EAX un 0.

El programa principal llama al procedimiento *copiaDigitos* indicándole que procese *cadenaOrigen* y que deje el resultado del procesamiento en *cadenaDestino*. Finalmente, almacena el número de dígitos copiados en la variable *numDigitos*.

Datos adicionales Dirección de comienzo de la sección de datos: 00404000
Valor de ESP al inicio del programa: 0012FFC4

Contesta a las preguntas relativas a este programa que indican a continuación.

— ¿Qué instrucción es necesaria en el hueco (—**1**—) del listado? **0,5**

```
add esp, 8
```

— ¿Qué instrucciones son necesarias en el hueco (—**2**—) del listado? **0,75**

```
inc  edx
mov  [edi], bl
inc  edi
```

— Escribe el código necesario en el hueco (—**3**—) del listado. **0,75**

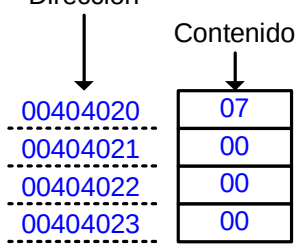
```
  mov  eax, 1
  jmp  SHORT sigue
_noEsDigito:
  mov  eax, 0
```

— Determina los valores que toma el registro EBP en los siguientes momentos de la ejecución del programa: **0,75**

En el ámbito de ejecución de copiaDigitos, es decir, cuando EBP apunta al marco de pila de copiaDigitos:
EBP = 0012FFB4

En el ámbito de ejecución de esDigito, es decir, cuando EBP apunta al marco de pila de esDigito:
EBP = 0012FF98

— A la derecha se muestra la representación en memoria de los cuatro bytes de la variable *numDigitos*. Escribe a la izquierda de cada byte su dirección de memoria y dentro de cada uno de ellos su contenido justo después de la ejecución de la instrucción marcada con ♥♥♥. Escribe todos los valores en hexadecimal. **0,75**



— Determina el máximo valor que alcanzará el registro ESI durante la ejecución del procedimiento *copiaDigitos*. **0,5**

```
0040400F
```

— Codifica la instrucción "`mov esi, [ebp+12]`", marcada en el programa con el símbolo ♠♠♠. **0,5**

```
8B 75 0C
```

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

Examen de Arquitectura de Computadores. (Telemática.) **Parcial de diciembre: 12-12-2007**

A continuación se muestra el listado de un programa C muy simple, que solamente ejecuta una sentencia condicional:

```
int A=5, B=10, C=15;   // Variables globales

main()
{
      // Sentencia condicional
      if (C < 20)
          A = A+B;
}
```

— Escribe el conjunto de instrucciones ensamblador que generaría el compilador de C al compilar la sentencia condicional del programa anterior.

1

```
  cmp  [C], 20
  jge  sigue
  mov  eax, [A]
  add  eax, [B]
  mov  [A], eax
sigue:
```

Un determinado procedimiento recibe tres parámetros a través de la pila y utiliza tres variables locales, todos ellos (los parámetros y las variables) de 32 bis. Los parámetros, a los que nos referiremos en este ejercicio como *par1*, *par2* y *par3*, se ubican en el marco de pila en este orden, es decir, primero *par1*, luego *par2* y finalmente *par3*. las variables locales, a las que llamaremos *var1*, *var2* y *var3*, se ubican en el marco de pila también en este orden, o sea, primero *var1*, después *var2* y finalmente *var3*. Según esta ubicación el primer elemento del marco de pila del procedimiento será *par1* y el último *var3*. El marco de pila del procedimiento se construye de la forma estándar. Teniendo en cuenta esta información, contesta a las preguntas A y B.

— A) Escribe un bucle controlado por contador que sume el valor 5 a la variable local *var3* tantas veces como indique *par1*. Usa la instrucción *loop* e inicializa el registro ECX de la forma apropiada. *par1* y *var3* deben direccionarse de la forma estándar.

1

```
  mov  ecx, [ebp+16]
bucle:
  add  [ebp-12], DWORD PTR 5
  loop bucle
```

— B) Escribe las instrucciones necesarias para destruir el marco de pila del procedimiento de la forma estándar. Ten en cuenta que el propio procedimiento es el encargado de destruir sus parámetros.

0,5

```
mov esp, ebp
pop ebp
ret 12
```

— Contesta a las siguientes preguntas breves:

1

Escribe las instrucciones que consideres oportunas para convertir un dato de 8 bits interpretado con signo en un dato de 32 bits. El dato origen se encuentra en el registors BL. El dato convertido debe dejarse en el registro EDX.

```
movsx  edx, bl
```

Escribe una sola instrucción que convierta una letra minúscula almacenada en el registro AL en mayúscula (datos: ASCII('A') = 41h; ASCII('a') = 61h)

```
and al, 0DFh
```

Escribe una secuencia de instrucciones que lean un dato del puerto de E/S 300h y lo almacenen en la variable *var*, de tipo byte, definida en la sección de datos.

```
mov dx, 300h
in  al, dx
mov [var], al
```

Indica cuál es el número de excepción que corresponde a la exceción de modo traza del procesador

1

A

— Indica cuáles son los dos modos de ejecución habitualmente ofrecidos por las CPUs para proporcionar dos niveles de privilegios de ejecución diferentes. Describe cómo se comporta la CPU en cada uno de estos modos.

0,75

Modo supervisor

Todas las instrucciones, incluidas las potencialmente peligrosas, pueden ser ejecutadas sin restricción alguna.

Modo usuario

- **Las instrucciones normales se pueden ejecutar sin restricción alguna.**
- **Las instrucciones potencialmente peligrosas son "examinadas" antes de su ejecución. Si se detecta que violan algún mecanismo de protección se impide su ejecución, señalizándose el problema mediante una excepción. En el caso contrario, se ejecutan.**

— Define el concepto de jerarquía de memoria.

0,75

Se trata de combinar diferentes tecnologías de memoria, rápidas y grandes, de modo que:

- **La memoria rápida contenga en cada momento la parte del programa o programas a los que se está accediendo con mayor frecuencia.**
- **La memoria lenta contenga el resto del programa o programas en ejecución.**

— Indica cuál o cuáles de las siguientes afirmaciones son ciertas. contesta NINGUNA si creses que ninguna lo es.

A) El modo protegido del 80286 funciona como una arquitectura de 16 bits.

B) Un programa realizado para una plataforma Windows podrá ejecutarse sin problemas sobre una plataforma Linux y viceversa, siempre que la arquitectura del computador sobre la que se ejecuten esas plataformas sea la IA-32.

C) En los PCs ya no se utiliza el lenguaje máquina de 16 bits.

D) En el diseño de una nueva evolución de un procesador perteneciente a la familia IA-32, no sería aceptable introducir cambios en los códigos de operación utilizados por las instrucciones aritmético-lógicas definidas en dicha familia.

0,5

A, D