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

Examen de Arquitectura de Computadores. (Telemática.) **Parcial de diciembre: 13-12-2006**

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

.DATA
cadOrg      DB  "Hola-Mundo", 0
cadDes      DB  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
lon_cadDes  DD  15
exito       DB  ' '

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

  ; Recuperar parámetros ---------------------------------
  ; ESI debe apuntar a la cadena origen, EDI a la cadena destino,
  ; y EBX a la variable que contiene la longitud de la cadena destino


  (--1--)

  xor   edx, edx     ; DL auxiliar para mover caracteres
  mov   ecx, [ebx]   ; ECX contiene el tamaño de la cadena destino

  ; Poner a 0 el contador de caracteres de la cadena destino
  mov   DWORD PTR [ebx], 0  ◄◄◄

  ; Copiar la cadena hasta alcanzar el terminador
  ; o hasta que se llene la cadena destino
bucle:
  cmp  [ebx], ecx
  je   fuera1
  cmp  BYTE PTR [esi], 0
  je   fuera2


  (--2--)


  jmp  bucle
```

```
fuera2:
  ; Copiar el terminador de cadena
  mov  dl, [esi]
  mov  [edi], dl
  inc  DWORD PTR [ebx]
  ; La cadena se ha compiado completamente
  mov  eax, 1
  jmp  sigue

fuera1:
  ; La cadena destino está llena,
  ; así que no se ha copiado completamente
  mov  eax, 0
sigue:

  ; Restauración de registros y retorno
  pop ecx
  pop edx
  pop ebx
  pop edi
  pop esi
  pop ebp
  ret  12  ♠♠♠
copyCad ENDP

inicio:
  ; Llamar a copyCad
  push  OFFSET cadOrg
  push  OFFSET cadDes
  push  OFFSET lon_cadDes
  call  copyCad

  ; Actualizar la variable exito


  (--3--)


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

El listado anterior corresponde a un programa cuyo objetivo es copiar una cadena de caracteres de la sección de datos en otra cadena, también de la sección de datos. Las cadenas manejadas se encuentran terminadas con el número cero.

A

Para llevar a cabo el procesamiento requerido, el programa utiliza el procedimiento *copyCad*. Este procedimiento recibe tres parámetros a través de la pila: 1) la dirección de la cadena que se quiere copiar (cadena origen), 2) la dirección de la cadena en la que se desea realizar la copia (cadena destino), y 3) la dirección de una variable que indica el tamaño de la cadena destino.

El procedimiento *copyCad* utiliza la variable que indica el tamaño de la cadena destino como mecanismo de seguridad. Primero *copyCad* copia el contenido de esta variable en un registro, ya que luego esta variable se va a modificar. Entonces, antes de copiar cada nuevo carácter de la cadena origen a la cadena destino, *copyCad* chequea el valor de dicho registro. Si se alcanza el valor almacenado en este registro, se termina la copia de caracteres y así se evita que se continúen copiando caracteres en la cadena destino, más allá del espacio de almacenamiento proporcionado por ésta. *copyCad* utiliza la variable que indica el tamaño de la cadena destino para otro cometido que se indica a continuación. *copyCad* pone esta variable a 0 y, después, durante la copia de caracteres, *copyCad* incrementa el valor de esta variable, de modo que cuando se termine la ejecución del procedimiento, esta variable contendrá el número de caracteres copiados.

copyCad devuelve en el registro EAX un 1 si tiene éxito en la copia de la cadena, lo cual significa que ha podido copiar completamente la cadena origen en la cadena destino, incluido el terminador. En el caso contrario, devuelve un 0, lo que indica que el tamaño reservado para la cadena destino es menor que el tamaño de la cadena origen.

El programa principal llama al procedimiento *copyCad* indicándole que copie `cadOrg` en `cadDes`. Como puede observarse en la sección de datos del programa, la variable que contiene la longitud de la cadena destino es `lon_cadDes`. El programa principal utilizando el valor devuelto por el procedimiento, actualiza la variable `exito`. Esta variable debe cargarse con el carácter 'S' si *copyCad* ha podido copiar toda la cadena origen en la cadena destino y con el carácter 'N' en el caso contrario.

Contesta a las preguntas que se proporcionan después del listado. Allí donde sea pertinente, ten en cuenta los comentarios del programa en tus respuestas.

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

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

0,75

```
mov  dl, [esi]
mov  [edi], dl
inc  DWORD PTR [ebx]
inc  esi
inc  edi
```

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

0,75

```
mov   esi, [ebp+16]
mov   edi, [ebp+12]
mov   ebx, [ebp+8]
```

— Escribe las instrucciones necesarias en el hueco (—**3**—) del listado. El objetivo de estas instrucciones es actualizar la variable `exito` en la forma indicada en la descripción del programa. Para ello, utiliza todas las etiquetas que consideres oportunas.

0,75

```
  cmp   eax, 0
  je    exito_no
  ; La cadena se ha compiado con éxito
  mov   [exito], 'S'
  jmp   sigue2
exito_no:
  ; La cadena no se ha compiado completamente
  mov   [exito], 'N'
sigue2:
```

— Determina el valor contenido en la dirección de memoria 0012FFBC durante la ejecución del procedimiento *copyCad*. Se pregunta por una única posición de memoria de 8 bits, debido a ello tienes que contestar con dos dígitos hexadecimales.

0,75

```
0B
```

— Determina el valor más grande que alcanza el registro EDI durante la ejecución del procedimiento *copyCad*. Contesta con 8 dígitos hexadecimales.

0,75

```
00404015
```

— Codifica la instrucción "`mov DWORD PTR [ebx], 0`", marcada en el listado con el símbolo ◄◄◄. Contesta en hexadecimal.

0,75

```
C7 03 00 00 00 00
```

— ¿Qué valor contendrá el registro ESP justo antes de que se ejecute la instrucción "`ret 12`", marcada en el listado con el símbolo ♠♠♠? Contesta con 8 dígitos hexadecimales.

0,5

```
0012FFB4
```

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

Examen de Arquitectura de Computadores. (Telemática.) **Parcial de diciembre: 13-12-2006**

A continuación se muestra el listado de un programa C muy simple. En este programa se define la función *esImpar()*, que recibe un número entero sin signo como parámetro y devuelve como resultado un 1, si el número que recibe como parámetro es impar y un 0, si no lo es. La función *main()* llama a la función *esImpar()* para determinar si su variable global C es impar y almacena el resultado en la variable D.

```
unsigned int esImpar( unsigned int );  // Prototipo

unsigned int A=1, B=3, C=0, D=0;       // Variables globales

main()
{
      C = A+B;  ♥♥♥

      D = esImpar( C );  ♦♦♦
}

unsigned int esImpar( unsigned int r )
{
      unsigned int aux;

      aux = r & 1;

      return aux;
}
```

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 "C = A+B", marcada con el símbolo ♥♥♥ en el listado. **0,75**

```
mov  eax, [A]
add  eax, [B]
mov  [C], eax
```

— Escribe el conjunto de instrucciones ensamblador que generaría el compilador de C al compilar la sentencia "D = esImpar(C)", marcada con el símbolo ♦♦♦ en el listado. **0,75**

```
push [C]
call esImpar
add  esp, 4
mov  [D], eax
```

— Tres procedimientos A, B y C se ejecutan de forma anidada, de modo que C se ejecuta anidado en B y éste, a su vez, anidado en A. Se sabe que cada procedimiento recibe un parámetro de 32 bits a través de la pila y reserva espacio para una variable local, también de 32 bits. Ninguno de estos procedimientos salvaguarda en la pila ningún registro, salvo el EBP. Todos estos procedimientos construyen el marco de pila de la forma estándar. Antes de que el programa principal introduzca el parámetro para llamar al procedimiento A, el valor del registro ESP es 0012FFC4. Después A llama a B y éste a C. Se desea saber el rango de direcciones que ocupará en la pila el parámetro pasado al procedimiento C. (Ejemplo de respuesta: 0012FFC0 – 0012FFC2.) **0,75**

0012FFA0 – 0012FFA3

— Contesta a las siguientes preguntas breves: **0,75**

¿Por qué resulta muy problemático introducir cambios en la arquitectura de un determinado modelo de computador?

Porque los programas realizados para la arquitectura original dejarían de funcionar en la arquitectura modificada.

¿Qué nombre reciben los dos tipos de lenguaje máquina soportados por la arquitectura IA-32?

Lenguaje máquina de 16 bits y lenguaje máquina de 32 bits.

Escribe una secuencia de instrucciones que envíen el dato 50h al puerto 2000h de E/S.

```
mov dx, 2000h
mov al, 50h
out dx, al
```

A

— Indica cuáles son los tres mecanismos proporcionados por una CPU para dar soporte a las transferencias de control al sistema operativo. Indica brevemente el objetivo de cada uno de estos mecanismos.

0,75

Interrupciones:

Se trata de una señal enviada desde un periférico a la CPU, con objeto de que ésta abandone el programa en curso y ejecute una rutina de tratamiento del periférico. Dicha rutina formará parte del sistema operativo.

Excepciones:

Su objetivo es provocar una transferencia de control al SO cuando se alcanza una determinada condición de estado (normalmente un error) durante la ejecución de una instrucción.

Llamadas al sistema:

Su objetivo es que un programa ejecute una rutina del sistema operativo, con objeto de que dicha rutina proporcione un servicio al programa.

— Explica los conceptos de localidad espacial y localidad temporal en el acceso a las instrucciones y datos de un programa.

0,75

Localidad espacial:
Cuando un programa accede a una instrucción o a un dato, existe una elevada probabilidad de que instrucciones o datos cercanos sean accedidos pronto.

Localidad temporal:
Cuando un programa accede a una instrucción o a un dato, existe una elevada probabilidad de que esa misma instrucción o dato vuelva a ser accedido pronto.

— Indica cuál o cuáles de las siguientes afirmaciones son ciertas relativas a la arquitectura IA-32, contesta NINGUNA si creses que ninguna lo es.

A) La excepción de doble falta es de tipo fallo.

B) Las instrucciones de E/S se ejecutarán sin problemas (es decir, sin que generen excepciones de protección) si el valor numérico del CPL es superior al del IOPL. (Por ejemplo, CPL=3 e IOPL=2).

C) Las interrupciones enmascarables son recibidas por la CPU a través de la línea INTR.

D) Cuando se generan excepciones de tipo *trap*, el sistema se prepara para retronar a la instrucción siguiente a la que provocó la excepción.

0,5

C, D