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

Examen de Arquitectura de Computadores. (Telemática.) **Convocatoria de febrero, primera parte: 03-02-2009**

Se ha diseñado un programa en ensamblador cuyo objetivo es ordenar un vector de números enteros positivos de mayor a menor.

El programa tiene en la sección de datos 3 elementos:

- Un vector, llamado origen, que contiene una serie de números positivos desordenados. Cada elemento de este vector es de tipo doble palabra.
- Otro vector, llamado destino, que albergará los mismos números contenidos en el vector anterior, pero ordenados de mayor a menor. Inicialmente este vector está inicializado con todos sus elementos a cero. Cada elemento de este vector es de tipo doble palabra.
- Una variable de tipo byte que contiene la longitud de los dos vectores anteriores.

El algoritmo utilizado para realizar la ordenación es ineficiente pero sencillo. La idea consiste en recorrer el vector origen tantas veces como elementos tenga éste. En cada uno de estos recorridos se determina qué elemento es el mayor, y se copia al otro vector. Para que en el siguiente recorrido del vector origen no se determine que el mayor elemento es el mismo que obtuvimos antes, éste se "machaca" poniendo un 0 en su lugar. El elemento más grande que se localice en el primer recorrido ocupará la posición 0 del vector destino. El que se localice en el segundo recorrido, ocupará la posición 1, y así sucesivamente. De este modo, al finalizar la ejecución del programa, el vector origen quedará con todos sus elementos a 0 y el vector destino tendrá los mismos números que tenía inicialmente el vector origen, pero ordenados de mayor a menor. En el caso particular del programa mostrado más adelante, tenemos dos vectores de 7 elementos cuyo estado inicial es el siguiente:

origen	3	5	10	6	8	7	9
destino	0	0	0	0	0	0	0

Tras realizar el primer recorrido por el vector origen, se determina que el mayor elemento es el 10, que se copiará a la primera posición del vector destino. A su vez, el 10 se elimina del vector origen para no volver a ser elegido el elemento mayor, con lo que los vectores quedarían en el estado siguiente:

origen	3	5	0	6	8	7	9
destino	10	0	0	0	0	0	0

En un nuevo recorrido por el vector origen, se determinaría que ahora el mayor es el 9. Se volvería a escribir un 0 en esta posición, copiando dicho valor en la siguiente posición del vector destino.

origen	3	5	0	6	8	7	0
destino	10	9	0	0	0	0	0

Una vez finalizado el proceso, se habrán retirado todos los elementos del vector origen, y se habrán ido copiando ordenados de mayor a menor al vector destino, quedando ambos como se especifica a continuación:

origen	0	0	0	0	0	0	0
destino	10	9	8	7	6	5	3

Para realizar este proceso, es necesario utilizar dos bucles anidados. En cada iteración del bucle exterior se determinará el elemento más grande presente en el vector origen y se copiará en la primera posición vacía del vector destino. Para poder hacer este cálculo, en cada iteración habrá que recorrer, mediante un bucle interior, el vector origen completo para hallar su máximo. El criterio de para del bucle exterior será la condición de haber rellenado el vector destino (para apuntar a cada una de las posiciones del vector destino utilizaremos un registro como índice. Cuando este índice sea 7, es que el vector ya está lleno). Para controlar el bucle interior se utilizará la instrucción *loop* que utiliza siempre el registro ECX como contador.

El uso de los registros en el programa principal es el siguiente:

- eax: índice utilizado para direccionar elementos de la cadena origen
- ebx: índice utilizado para direccionar elementos de la cadena destino
- ecx: tamaño de los arrays
- edx: guardar el máximo encontrado hasta el momento en el vector origen
- esi: Guardar la posición del máximo encontrado hasta el momento dentro del vector origen

Una vez que se haya finalizado un recorrido por el vector origen en busca de su elemento máximo, tendremos en el registro edx el valor de éste, y en esi la posición que ocupa este máximo dentro del vector. Para escribir este máximo en la posición correspondiente del vector destino y poner a 0 la posición que ocupaba en el vector origen utilizamos un

procedimiento auxiliar. Dicho procedimiento recibe 4 parámetros, descritos a continuación **en orden de apilación**.

1. Dirección de comienzo del vector origen
2. Dirección de comienzo del vector destino
3. Posición del máximo en el vector origen
4. Posición en la que se ha de escribir el máximo en el vector destino

A continuación se muestra el listado correspondiente a este programa debidamente comentado.

```
.386
.MODEL FLAT, stdcall

ExitProcess PROTO, :DWORD

.DATA
	origen  DD 3, 5, 10, 6, 8, 7, 9
	destino DD 7 DUP(0)
	tamanio DB 7

.CODE

CopiaMaximo PROC
	; Creación del marco de pila
	push ebp
	mov ebp, esp

	; Salvaguarda de los registros utilizados por el procedimiento
	push eax
	push ebx
	push ecx
	push edx
	push esi

	; Guardar los parámetros en registros
	mov eax, [ebp+8]    ◄◄◄
	mov ebx, [ebp+12]
	mov ecx, [ebp+16]
	mov edx, [ebp+20]

	; Guardo el máximo encontrado en el programa principal
	; en la siguiente posición libre del vector destino
	mov esi, [edx+ebx*4]
	mov [ecx+eax*4], esi   ♠♠♠

	; Pongo un 0 en el lugar donde estaba antes el máximo
	mov [edx+ebx*4], DWORD PTR 0

	;Restauración de registros y retorno de la función
	( -- 3 -- )

CopiaMaximo ENDP


inicio:

	; Los registros que harán de índice, inicialmente se les da un
	; valor 0
	mov eax, 0
	mov ebx, 0

	bucleExterior:

	; Guardar en ECX el número de iteraciones a realizar en el
	; bucle exterior (debe ser igual al número de elementos de los
	; arrays). Utilizar la variable global que contiene este valor
	( -- 1 -- )

	; Si se han copiado ya tantos elementos en el vector destino
	; como tamaño tienen los arrays, hemos acabado
	cmp ebx, ecx
	je salir

	; Inicializar los registros eax y edx a 0 para la nueva
	; iteración del bucle exterior
	mov edx, 0
	mov eax, 0

	bucleInterior:

	; Comparo el máximo encontrado hasta el momento con el
	; siguiente elemento del vector origen
	cmp edx, [origen+eax*4]

	; Si el elemento que acabo de examinar es mayor que el máximo
	; encontrado hasta el momento, actualizo el valor del máximo
	; y su posición dentro del array

	( -- 2 -- )
```

```
      ; Incremento el índice utilizado para direccionar elementos
      ; del vector origen
      inc eax

      loop bucleInterior

      ; Llegados a este punto, el bucle interior calculó el
      ; valor del máximo y su posición dentro del vector

      ; La función CopiaMaximo se encargará de modificar los
      ; vectores
      push OFFSET origen  ; Dirección del vector origen
      push OFFSET destino ; Direccion del vector destino
      push esi            ; Posición del máximo en el vector origen
      push ebx            ; Próxima posición a escribir del vetor
                          ; destino

      call CopiaMaximo

      ; Incrementamos ebx para escibir en la siguiente posición
      ; del vector destino en la proxima iteracion
      inc ebx

      jmp bucleExterior
salir:

      push 0
      call ExitProcess

END inicio
```

También se sabe que el estado de los registros nada más comenzar la ejecución del proceso (antes de ejecutar ninguna instrucción) es el que se muestra en la tabla de la siguiente página.

Registro	Contenido (en hexadecimal)
EAX	00 00 00 00
EBX	7F FD C0 00
ECX	00 12 FF B0
EDX	7C 91 E4 F4
ESI	07 58 F9 F4
EDI	00 00 00 00
EIP	00 40 10 3A
ESP	00 12 FF C4
EBP	00 12 FF F0

Teniendo en cuenta toda esta información, responde a las siguientes preguntas:

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

```
movzx ecx, [tamanio]
```

0,75

— ¿Qué instrucciones serán necesarias en el hueco (-- **2** --) del listado? Puedes utilizar tantas etiquetas como consideres necesario.

1

```
  jae noActualizarMaximo
  mov edx, [origen+eax*4]
  mov esi, eax
noActualizarMaximo:
```

— ¿Qué instrucciones serán necesarias en el hueco (-- **3** --) del listado?

0,75

```
pop esi
pop edx
pop ecx
pop ebx
pop eax
pop ebp
ret 16
```

A

— ¿Qué valor tendrá el registro ESP en el momento en que se ejecute por primera vez la instrucción del procedimiento CopiaMaximo marcada con el símbolo ◄◄◄ ?

00 12 FF 98 — **0,5**

— En el momento en que se ejecute la instrucción citada en la pregunta anterior, ¿qué dato de 4 bytes estará almacenado a partir de la dirección 0012FFAC?

00 12 FF F0 — **0,5**

— Codifica la instrucción "`mov [ecx+eax*4], esi`", marcada con el símbolo ♠♠♠ en el procedimiento CopiaMaximo.

89 34 81 — **1**

— Responde brevemente a las siguientes cuestiones:

1

¿Cómo se llama el registro utilizado para almacenar los bits de estado y control de la CPU en la arquitectura IA32?

EFLAGS

¿Qué tipos de excepciones existen en la arquitecuta IA32?

Fallos, abortos y trampas

¿Qué información se almacena en el marco de pila de una función?

Sus parámetros, dirección de retorno, valor anterior del registro EBP y variables locales.

— Define los siguientes conceptos:

1

Arquitectura de computadores:
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 de un computador:

Conjunto de componentes físicos que conforman el ordenador, así como sus interrelaciones.

Jerarquía de memoria:
Es una técnica consistente en combinar memorias de diferentes tecnologías con el objetivo de construir un sistema de memoria rápido y grande.

— De las instrucciones mostradas a continuación, indica cuáles pueden ser ejecutadas cuando la CPU se encuentre en modo usuario. Si crees que no se podría ejecutar ninguna de ellas, responde NINGUNA.

A) in al, 20h

B) movsx, eax, bx

C) cli

D) ret

B, D — **0,5**

— Dibuja un esquema de cómo se traducen las direcciones virtuales a direcciones físicas. En él, deben aparecer todos los elementos (tanto software como hardware) implicados en el proceso de traducción.

0,75

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

Examen de Arquitectura de Computadores. (Telemática.) **Convocatoria de febrero, primera parte: 03-02-2009**

A continuación se muestra el listado de un programa escrito en C.

```
#include <stdio.h>

int vector[5]={3,5,4,6,8};
int global;

int Suma (int *v)
{
	int aux,i;

	aux=0;
	i=0;

	while (i<5){
		aux=aux+v[i];
		i++;
	}


	return aux;
}

main ()
{

	int * p;	//Instrucción 1
	p=vector;	//Instrucción 2


	global=Suma(p);	//Instrucción 3

	printf("La suma de los elementos del vector es %d\n",global);

}
```

Escribe en los huecos siguientes las instrucciones que generaría un compilador de C para traducir cada una de las instrucciones señaladas de la función "main".

Nota: En el código generado sólo se pueden utilizar las etiquetas de variables globales.

— Código generado para la "Instrucción 1"

```
sub esp, 4
```

0,75

— Código generado para la "Instrucción 2"

```
mov [ebp-4], OFFSET vector
```

0,75

— Código generado para la "Instrucción 3"

```
push DWORD PTR [ebp-4]
call Suma
add  esp, 4
mov  [global], eax
```

0,75