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

Examen de Arquitectura de Computadores. (Telemática) **Convocatoria de enero, primera parte: 24-01-2011**

Se dispone de un programa que utiliza listas para almacenar una colección de números enteros. Sobre una lista pueden realizarse dos operaciones: insertar y extraer. La estructura de las listas es de tipo FIFO (First In, First Out), es decir, el primer elemento que haya sido insertado en la lista también será el primero que se saque de ella. Cada vez que se realiza una extracción de la lista, se saca el elemento que ocupa el primer lugar, y luego se mueven todos los demás elementos adelantándolos una posición, de modo que el que antes era segundo pasa a ser primero, y así sucesivamente.

Estas listas se implementan mediante un *array* de elementos de tipo DOBLE PALABRA y cuyo tamaño máximo está fijado a 8 elementos. Adicionalmente, se tiene otra variable, también de tipo DOBLE PALABRA, que contiene el número de elementos que hay dentro de la lista en cada momento.

En el caso particular del programa presentado en la página siguiente, se parte de una lista vacía declarada en la sección de datos y de nombre *lista*. La variable utilizada para llevar la cuenta del número de datos que se almacena también está declarada en la sección de datos y su nombre es *contador*. Sus respectivos contenidos se muestran a continuación.

lista	0	0	0	0	0	0	0	0

contador	0

Si se introduce un ‘45’ en la lista, su estado pasa al mostrado en la siguiente figura.

lista	45	0	0	0	0	0	0	0

contador	1

Y tras realizar una nueva inserción, esta vez el número ‘12’, quedaría del siguiente modo:

lista	45	12	0	0	0	0	0	0

contador	2

Se podría seguir rellenando la lista del mismo modo, manteniendo siempre actualizado el contador, hasta que se hayan insertado 8 elementos. Si en el estado en que se encuentra la lista en la figura anterior se realizara una extracción, SIEMPRE se extraería de la lista el primero de los elementos insertados, es decir, el ‘45’. Tras realizar una operación de extracción, la lista quedaría como se muestra a continuación.

lista	12	0	0	0	0	0	0	0

contador	1

El contador se actualizó, se eliminó el ‘45’ de la lista y el que era antes el segundo elemento ocupa ahora el primer lugar, y será el elemento que se extraerá de la lista cuando se produzca la siguiente operación de extracción.

Para manejar las listas, el programa cuenta con dos procedimientos, encargados de realizar las operaciones de inserción y extracción. Dichos procedimientos se detallan a continuación:

- Procedimiento “insertar”: Se utiliza para insertar un elemento en una lista. Recibe tres parámetros a través de la pila (todos de 32 bits). Los parámetros, en orden de apilación, son los siguientes:
  1. Dirección de la lista a utilizar (paso por referencia).
  2. Dirección del contador asociado a la lista a utilizar (paso por referencia).
  3. Valor a insertar

  Este procedimiento devuelve un valor en el registro EAX. Dicho valor es 0 si la inserción se realizó correctamente. Si por problemas de espacio no fue posible insertar el elemento solicitado en la lista, el valor devuelto en el registro EAX es 1.

  El procedimiento se encarga de incrementar en una unidad el contenido del contador que se le pase como segundo parámetro (si la inserción tuvo éxito).

- Procedimiento “extraer”: Se utiliza para extraer el primer elemento de una lista. Recibe dos parámetros a través de la pila (ambos de 32 bits). Los parámetros, en orden de apilación, son los siguientes:
  1. Dirección de la lista a utilizar (paso por referencia).
  2. Dirección del contador asociado a la lista a utilizar (paso por referencia).

  Este procedimiento devuelve el contenido del primer elemento de la lista en el registro EAX. Adicionalmente, desplaza todos los elementos de la lista una posición hacia adelante, de modo que el elemento que antes ocupaba la posición N, pasará a ocupar la posición N-1. Si la lista está vacía devuelve 0.

  El procedimiento se encarga de decrementar en una unidad el contenido del contador que se le pase como segundo parámetro.

· **La sección de datos del programa comienza a partir de la dirección 0x00404000**

· **En contenido del registro ESP al comienzo de la ejecución es 0x0012FFC4**

```
.386
.MODEL flat, stdcall

ExitProcess PROTO, :DWORD

.DATA

lista   DD 8 DUP (0)
contador DD 0
variable DD 0

.CODE

insertar PROC
      push  ebp
      mov   ebp, esp

      push  ebx    ;Salvaguarda de registros
      push  ecx
      push  edx
      push  esi

      ;Lectura de parámetros
      ;EBX: valor   ECX: Dirección contador    ESI: Dirección lista

      ( -- 2 -- )

      mov   edx, [ecx]    ;Guardamos en EDX el contenido del contador
      xor   eax, eax

      cmp    edx, 8
      jae    error
      ;Si no saltó, todo fue bien (num. de elementos < 8)

      ;Insertar el elemento en la posición correspondiente
      ;Incrementar el contador pasado como parámetro
      ;Poner un 0 en EAX

      ( -- 3 -- )

      jmp   salir
error:
      mov   eax, 1
salir:
      pop   esi
      pop   edx
      pop   ecx
      pop   ebx

      pop   ebp
      ret
insertar ENDP

extraer PROC
      push  ebp
      mov   ebp, esp

      push  ebx    ;Salvaguarda de registros
      push  ecx
      push  edx
      push  esi

      ;Lectura de parámetros
      ;ECX: Dirección contador    ESI: Dirección lista
```

No se muestran estas instrucciones

```
      ;Guardamos en EDX el contenido del contador
      ;Se utilizará como contador en el bucle
      mov   edx, [ecx]
      xor   ebx, ebx      ;EBX: Registro auxiliar

      cmp   edx, 0
      je    error
      ;Si llegó aquí, es que hay elementos

      mov   eax, [esi] ;Dejamos en EAX el valor a devolver
      mov   DWORD PTR [esi], 0

      ;Decrementamos el contador asociado a la lista
```

No se muestra esta instrucción

```
      ;Decrementamos EDX para que el bucle itere N-1 veces
      ;Siendo N el número de elementos que tenía la lista antes
      ;de realizar la llamada. En cada iteración mover un elemento
      ;distinto de la lista una posición hacia el comienzo
      dec   edx
```

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

Examen de Arquitectura de Computadores. (Telemática) **Convocatoria de enero, primera parte: 24-01-2011**

```
bucle:
      cmp   edx, 0
      je    finBucle

      ( -- 4 -- )

      jmp   bucle
finBucle:

      jmp   salir
error:
      mov   eax, 0
salir:
      pop   esi
      pop   edx
      pop   ecx
      pop   ebx

      pop   ebp
      ret
extraer ENDP

inicio:

      ( -- 1 -- )


final:
      push  0
      call  ExitProcess

END inicio
```

— Escribe las instrucciones correspondientes al hueco (-- **2** --) del listado, donde se realiza la lectura de los parámetros del procedimiento *insertar*.

0,5

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

— El hueco (-- **1** --) del programa se corresponde con el programa principal. Escribe a continuación las instrucciones necesarias para que, en el programa principal se realicen dos inserciones y una extracción. Se debe insertar en primer lugar el número 45. Si la inserción tuvo éxito, insertar a continuación el número 12. Por último, realizar una extracción de la lista y guardar el valor obtenido en la variable global *variable*.

1

```
push  OFFSET lista
push  OFFSET contador
push  45
call  insertar
add   esp, 12

cmp   eax, 1
je    final

push  OFFSET lista
push  OFFSET contador
push  12
call  insertar
add   esp, 12


push  OFFSET lista
push  OFFSET contador
call  extraer
add   esp, 8
mov   [variable], eax
```

— Escribe las instrucciones correspondientes al hueco (-- **4** --) del listado. Dicho hueco se corresponde con el cuerpo de un bucle tipo *while* en el procedimiento *extraer*. En cada iteración de dicho bucle se debe adelantar una posición a un elemento de la lista (lista[i]=lista[i+1]), y poner a 0 la posición que ocupaba el elemento que se ha desplazado (lista[i+1]=0). Además, se deben dejar preparados todos los registros con los valores apropiados para que se ejecute correctamente la siguiente iteración. Puedes utilizar el registro EBX de forma auxiliar si lo consideras necesario.

1

```
mov   ebx, [esi+4]
mov   DWORD PTR [esi+4], 0
mov   [esi], ebx
add   esi, 4
dec   edx
```

A

— Escribe las instrucciones correspondientes al hueco (-- **3** --) del listado, donde se inserta un elemento en la lista, se actualiza el contador y se prepara el registro EAX con su valor de retorno.

0,75

```
mov   [esi+edx*4], ebx
inc    DWORD PTR [ecx]
mov    eax, 0
```

— Determina cuál es el mínimo valor que alcanza el contenido del registro ESP durante la ejecución de este programa con el programa principal solicitado para el hueco (-- **1** --), donde se realizan dos inserciones y una extracción a la lista. Contestar en hexadecimal.

0,5

```
00 12 FF A0
```

— En un determinado instante, la lista del programa se encuentra en la situación indicada en la tercera figura de la primera página del examen, es decir, con un elemento de valor '45' ocupando la primera posición y un elemento de valor '12' ocupando la segunda (el resto de los elementos de la lista son '0'). Escribe el contenido que tendrán las posiciones de memoria indicadas en ese instante (en hexadecimal).

0,5

```
00 40 40 02:  00
00 40 40 03:  00
00 40 40 04:  0C
00 40 40 05:  00
```

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

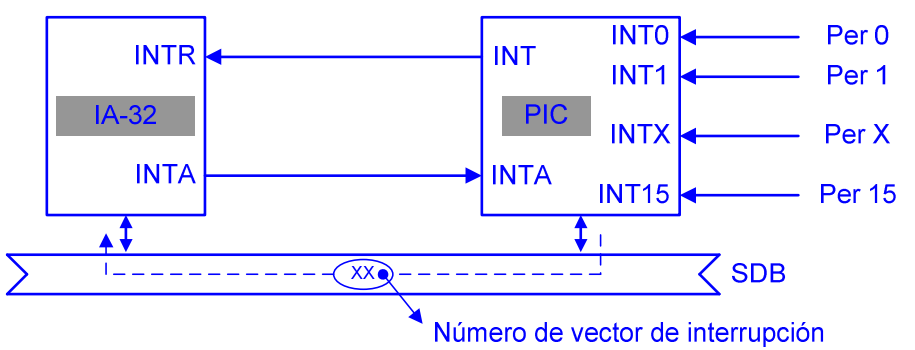
0,75

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.

— Dibuja el esquema de gestión de interrupciones en la arquitectura IA-32 mediante el dispositivo PIC e indica los 4 primeros pasos llevados a cabo por el PIC y el procesador para gestionar la petición de una interrupción por parte de un periférico.

1



1) El periférico solicita interrupción al PIC a través de una línea INTX.
2) El PIC pasa la solicitud de interrupción a la CPU a través de la línea INTR.
3) Si la CPU acepta la interrupción, se lo indica al PIC a través de la línea INTA.
4) El PIC envía a la CPU el número de vector de interrupción correspondiente al periférico que solicitó la interrupción. Cada línea INTX tiene asociado un número distinto, identificándose de esta forma a los diferentes periféricos. La asociación de números de vectores de interrupción a las líneas INTX es programable.

— Codifica la instrucción mov [lista+esi*4], 33. La etiqueta lista se corresponde a la variable global del listado del programa de las preguntas anteriores. Contestar en hexadecimal

0,75

```
C7 04 B5 00 40 40 00 21 00 00 00
```

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

Examen de Arquitectura de Computadores. (Telemática) **Convocatoria de enero, primera parte: 24-01-2011**

— Indica los diferentes tipos de excepciones disponibles en la arquitectura IA-32 y define brevemente su cometido:

0,5

Fallos: Se utilizan para señaliar errores no catastróficos que pueden ser tratados sin que se pierda la estabilidad del sistema.

Abortos: Se utilizan para señalizar errores catastróficos que no permiten continuar la ejecución del sistema de una forma estable.

Trampas: Se utilizan para transferir el control a rutinas de depuración.

— Responde a las siguientes preguntas cortas:

0,75

Define el concepto de plataforma de ejecución:
Es el conjunto formado por el hardware de un computador y el sistema operativo que controla dicho hardware.

¿Qué elementos se guardan en el marco de pila de un procedimiento?
Sus parámetros, la dirección de retorno, el valor del registro EBP del procedimiento llamador y las variables locales al procedimiento.

Escribe la instrucción necesaria para copiar el contenido del registro AL, utilizado para guardar números enteros, en el registro EBX.

```
movsx ebx, al
```

A continuación se muestra el listado de un programa escrito en C. Dicho programa tiene definida una función encargada de sumar una cantidad a una variable. Al pasar la variable por referencia (se pasa su DIRECCIÓN, no su valor) las modificaciones realizadas por la función persisten una vez que se haya salido de ésta. De este modo, el primero de los printf presentes en el main escribirá un '40' por salida estándar, mientras que el segundo escribirá un '50'.

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

int dato=40;

void Suma (int * variable, int valor)
{
	int local; //Instrucción 2

	local=*variable;  //Instrucción 3

	local = local + valor;  //Instrucción 4

	*variable = local;
}

main()
{

	printf("Antes: %d\n", dato);


	Suma (&dato, 10); //Instrucción 1


	printf("Despues: %d\n", dato);


	_getch();
}
```

Nota: No utilizar nunca las etiquetas de parámetros o de variables locales.

Escribe las instrucciones generadas al compilar las siguientes instrucciones:

— Instrucción 1:

0,5

```
push 10
push OFFSET dato
call Suma
add  esp, 8
```

— Instrucción 2:

0,5

```
sub  esp, 4
```

Instrucción 3 (puedes utilizar los registros que necesites):

0,5

```
mov eax, [ebp+8]
mov ecx, [eax]
mov [ebp-4], ecx
```

Instrucción 4 (puedes utilizar los registros que necesites):

0,5

```
mov eax, [ebp-4]
add eax, [ebp+12]
mov [ebp-4], eax
```