Laboratorio de Sistemas Basados en Microprocesadores

Práctica 3: Diseño de programas utilizando C y ensamblador del 80x86

La empresa de videojuegos Feic SL se encuentra en la fase de desarrollo de un juego basado en el famoso "*MasterMind*". El juego consiste en adivinar un número secreto de 4 dígitos en intentos sucesivos (posibles valores de cada dígito entre 0 y 9). Cada uno de los dígitos del número secreto debe ser generado aleatoriamente y el número secreto no puede contener dígitos repetidos. En cada intento, el usuario introducirá un número de 4 dígitos. El programa comparará el número secreto con el intento tecleado por el usuario y ofrecerá la siguiente información por pantalla: cuántos dígitos coinciden y están en su posición (número de aciertos) y cuántos dígitos coinciden, pero no están en su posición (número de semiaciertos).

Debido a restricciones en cuanto a la rapidez de ejecución y al tamaño ocupado en memoria, el juego debía realizarse en dos lenguajes diferentes: lenguaje C para el programa principal y ensamblador del 80x86 para las funciones internas. El desarrollo fue encargado al programador Zote Cenutrio Pazguato, debido a su gran experiencia y rapidez en la codificación en ambos lenguajes. Lamentablemente para la empresa, cuando ya tenía la parte del lenguaje C terminada y sólo le faltaba codificar las funciones internas en ensamblador, Zote ganó el Euromillón, permaneciendo ilocalizable desde entonces.

En esta práctica deben codificarse en ensamblador las funciones llamadas desde el código ya terminado en C (proporcionado en el Anexo II) para completar el desarrollo del juego. El programa principal escrito en C (pract3.c) hace llamadas a diferentes funciones, que deben ser desarrolladas en ensamblador en dos ficheros (módulos) diferentes. El alumno deberá realizar el correspondiente análisis de los requisitos expuestos y realizar su implementación. Cada una de las funciones desarrolladas tiene el mismo peso en la nota final de la práctica.

Módulo 1: pract3a.asm

En el módulo pract3a.asm se deben desarrollar en ensamblador las siguientes funciones, que serán llamadas desde el programa principal en C de acuerdo a los siguientes prototipos:

1.- unsigned int comprobarNumeroSecreto(unsigned char* numero);

Devuelve un 1 si "numero[]" contiene algún digito repetido, un cero en caso contrario ("numero[]" es un array de 4 posiciones, cada posición del array contiene un dígito del número secreto).

2.- void rellenarIntento(unsigned int intento, unsigned char* intentoDigitos);

Rellena el array de 4 posiciones "intentoDigitos[]", de forma que en cada posición del array almacena el dígito correspondiente del número de 4 cifras "intento". Por ejemplo, si intento es el número 4567, la función debe rellenar el array de la siguiente forma: intentoDigitos[0] = 4; intentoDigitos[1] = 5; intentoDigitos[2] = 6; intentoDigitos[3] = 7.

Módulo 2: pract3b.asm

En el módulo pract3b.asm se deben desarrollar en ensamblador las siguientes funciones, que serán llamadas desde el programa principal en C de acuerdo a los siguientes prototipos:

1.- unsigned int calcularAciertos(unsigned char* numSecreto, unsigned char* intentoDigitos);

Devuelve el número de dígitos acertados existente en el array de 4 posiciones "intentoDigitos[]" con respecto al array de 4 posiciones "numSecreto[]". Un dígito de "intentoDigitos[]" se considera acertado si coincide el valor y la posición del mismo con respecto a los de "numSecreto[]". Por ejemplo, si numSecreto[] = {1,2,3,4} e intentoDigitos[] = {5,6,3,4}, la función debe devolver el valor 2.

2.- unsigned int calcularSemiaciertos(unsigned char* numSecreto, unsigned char* intentoDigitos);

Devuelve el número de dígitos semi-acertados existente en el array de 4 posiciones "intentoDigitos[]" con respecto al array de 4 posiciones "numSecreto[]". Un dígito de "intentoDigitos[]" se considera semi-acertado si coincide el valor pero no la posición del mismo con respecto a los de "numSecreto[]". Por ejemplo, si numSecreto[] = {1,2,3,4} e intentoDigitos[] = {3,1,7,4}, la función debe devolver el valor 2.

Notas:

- El compilador precede todas las referencias externas con el carácter "_", por lo que es necesario que todas las funciones desarrolladas en ensamblador comiencen con el mismo (ejemplo: `_calcularAciertos`)
- Todas las funciones desarrolladas en ensamblador deben ser declaradas como PUBLIC para poder ser llamadas desde el programa principal
- No se permite la declaración de variables globales en ensamblador. De hecho, cada módulo desarrollado en ensamblador contendrá **exclusivamente** un solo segmento de código, que debe comenzar/finalizar con las siguientes líneas:

```
<nombre módulo> SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS: <nombre módulo>
…
…
<nombre módulo> ENDS

END
```

- El programa desarrollado en C (pract3.c) debe ser compilado en modelo LARGE (se proporciona un *makefile* en el Anexo I). Por tanto, las funciones desarrolladas en ensamblador serán FAR.

ENTREGA DE LA PRÁCTICA: Fecha y contenido

Se deberá subir a Moodle un fichero zip que contenga los ficheros fuentes de los programas y el fichero makefile. Sólo podrá ser subido por uno de los miembros de la pareja.

Los ficheros a entregar deberán contener en la cabecera los nombres de los autores y el identificador de la pareja. Así mismo, el código de los ficheros entregados deberá estar correctamente tabulado y comentado. La falta de comentarios o la baja calidad de éstos, será calificada negativamente.

El límite de fecha de subida de los ficheros, para cada grupo es el siguiente:

Grupos del Martes: 16 de Abril a las 23:55h

Grupos del Miércoles: 17 de Abril a las 23:55h

Grupos del Viernes: 19 de Abril a las 23:55h

Anexo I: Compilación de un proyecto desarrollado en C y ensamblador

En esta práctica disponemos de 3 ficheros, el programa principal escrito en C (pract3.c) y los 2 módulos que contienen las funciones escritos en ensamblador (pract3a.asm y pract3b.asm).

Para realizar la compilación del programa en C se utilizará el compilador del TurboC (tcc). Para conocer todas las opciones de compilación que ofrece podemos ejecutar tcc sin parámetros dentro del DosBox. Los módulos en ensamblador se compilarán como en las prácticas anteriores, utilizando el tasm.

El programa en C debe ser compilado con la opción *–ml* (memory model large), mientras que los módulos en ensamblador deben ser ensamblados con la opción */ml* (case sensitivity on all symbols). Para generar el ejecutable final "pract3.exe" podemos utilizar el siguiente makefile (respetando los tabuladores):

```
all: pract3.exe

pract3.exe: pract3.obj pract3a.obj pract3b.obj

	tcc -v -ml -Lc:\compila\tc\lib pract3.obj pract3a.obj pract3b.obj

pract3.obj: pract3.c

	tcc -c -v -ml -Ic:\compila\tc\include pract3.c

pract3a.obj: pract3a.asm

	tasm /zi /ml pract3a,,pract3a

pract3b.obj: pract3b.asm

	tasm /zi /ml pract3b,,pract3b
```

Anexo II: Programa principal desarrollado en C

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

#define TRUE  1
#define FALSE 0
#define NUMEROINTENTOS 5

unsigned int comprobarNumeroSecreto(unsigned char* numero);
void rellenarIntento(unsigned int intento, unsigned char* intentoDigitos);
unsigned int calcularAciertos(unsigned char* numSecreto, unsigned char* intentoDigitos);
unsigned int calcularSemiaciertos(unsigned char* numSecreto, unsigned char*
intentoDigitos);

////////////////////////////////////////////////////////////////////////
///// ------------------------- MAIN ----------------------------- /////
////////////////////////////////////////////////////////////////////////

int main( void )
{

	int t;
	unsigned char numSecreto[4];
	unsigned char intentoDigitos[4];
	unsigned int numIntentos, intento, aciertos, semiaciertos, repetido, i;

	srand((unsigned) time(&t));

	do {
		for (i=0; i<4; i++)
			numSecreto[i] = rand() % 10;
		repetido = comprobarNumeroSecreto(numSecreto);
	} while (repetido == TRUE);

	numIntentos = 0;

	do {
		numIntentos++;
		do
		{
			printf("Introduzca intento %u [0000 - 9999]: ", numIntentos );
			scanf("%u", &intento);
		}
		while ( intento > 9999);

		rellenarIntento( intento, intentoDigitos );
		aciertos = calcularAciertos(numSecreto, intentoDigitos);
		semiaciertos = calcularSemiaciertos(numSecreto, intentoDigitos);
		printf("Numero de Aciertos: %u\t", aciertos);
		printf("Numero de Semiaciertos: %u\n", semiaciertos );

	} while ((aciertos != 4) && (numIntentos != NUMEROINTENTOS));

	if (aciertos == 4)
		printf("Combinacion correcta: HAS GANADO!!!\n");
	else
		printf("Numero de intentos excedido: HAS PERDIDO :(\n");

	printf("Numero secreto: %u%u%u%u\n", numSecreto[0], numSecreto[1], numSecreto[2],
numSecreto[3]);

	return 0;
}
```