Diseño Basado en Microprocesadores
Práctica 3
Programación en ensamblador x86 de 32 bits (II)

Índice

---

1. Ejercicios

1.1. Ejercicio 1

Escribe una funcion en ensamblador que extraiga los campos de fracción, exponente y signo de un número en punto flotante de precisión simple (un dato de tipo `float`). Un número en punto flotante de precisión simple es un dato de 32 bits. El campo de fracción ocupa los bits 0 a 22, el campo de exponente los bits 23 a 30 y el bit 31 es el bit de signo.

El prototipo de la función es:

```
int extraer_campos_float(float flotante,
                         unsigned int * fraccion,
                         unsigned char * exponente,
                         unsigned char * signo);
```

donde

`flotante` es el valor flotante simple del que hay que extraer sus diferentes campos.

`ptr_fraccion` es un puntero a una variable de tipo `unsigned int` donde se almacenará el valor del campo de fracción.

`ptr_exponente` es un puntero a una variable de tipo `unsigned char` donde se almacenará el valor del campo de exponente.

`ptr_signo` es un puntero a una variable de tipo `unsigned char` donde se almacenará el valor del bit de signo.

La función retorna 1 si la operación se realizó con éxito y 0 si alguno de los punteros es nulo.

Aunque el primer argumento de la función sea un número en punto flotante podremos seguir accediendo a él en la posición [EBP + 8]. Además, de cara a extraer los campos de bits indicados, podemos obviar que en origen se trate de un número en punto flotante y tratarlo mediante las instrucciones de enteros que conocemos.

Puedes usar el convertidor en https://www.h-schmidt.net/FloatConverter/IEEE754.html para comprobar si los resultados que genera la función son correctos.

1.2. Ejercicio 2

Escribe una función en ensamblador que permita realizar la conversión de grados Celsius a Fahrenheit o viceversa. Las expresiones de conversión son:

$$T_{(^oF)} = \frac{9T_{(^oC)}}{5} + 32 \quad ; \quad T_{(^oC)} = \frac{5\left(T_{(^oF)} - 32\right)}{9}$$

El prototipo de la función es

```
int convertir_celcius_fahrenheit(int temperatura_entrada,
                                 int sentido conversion,
                                 int *temperatura_salida);
```

donde

`temperatura_entrada`: es la temperatura de partida, que puede estar en grados Celsius o Fahrenheit.

`sentido_conversion`: si vale 0 indica que la temperatura de entrada está en grados Celsius. En caso contrario, la temperatura de entrada está en grados Fahrenheit.

`temperatura_salida`: apunta a la variable donde almacenar el resultado de la conversión. En caso de que el resultado de aplicar las formulas de conversión tenga una parte fraccionaria, basta **truncar** el resultado hacia cero.

Prueba la función con temperaturas de entrada tanto positivas como negativas y que generen temperaturas de salida tanto positivas como negativas.

Mejora la función para que la temperatura resultante esté **redondeada** al número entero de grados más próximo al verdadero resultado.

1.3. Ejercicio 3

Escribe una función en ensamblador que calcule el histograma de un array de datos de tipo `unsigned char`, es decir, que cuente cuántas veces aparece cada valor posible para un dato de tipo `unsigned char` (de 0 a 255) en un array de datos de ese tipo. El prototipo de la función es:

```
int calcular_histograma(const unsigned char * ptr_datos,
                        unsigned int num_datos,
                        unsigned int *ptr_histograma);
```

donde

`ptr_datos` apunta al array con los datos de entrada.

`num_datos` indica el número de datos contenidos en el array apuntado por `ptr_datos`.

`ptr_histograma` apunta al array donde almacenar el histograma. Se supone que hay espacio para los 256 elementos del histograma.

La función retorna 1 si realizó su trabajo correctamente y 0 si alguno de los argumentos no es correcto.

Por ejemplo, tras ejecutar el siguiente fragmento

```
unsigned char array[10] = {1, 5, 4, 4, 4, 4, 1, 2, 1, 0};
unsigned int histograma[256];
calcular_histograma(array, 10, histograma);
```

el array `histograma` quedaría con los valores `{1, 3, 1, 0, 4, 1, 0, 0, 0, 0, 0,..., 0}` porque el 0 aparece 1 vez, el 1 aparece 3 veces, el 2 aparece 1 vez, el 3 no aparece ni una sola vez, el 4 aparece 4 veces, el 5 aparece una vez y el resto de valores entre 6 y 255 no aparecen ni una sola vez.