

Práctico 2 – Manipulación de la representación interna de datos en programas

Objetivos:

- Favorecer la comprensión de la representación interna de datos mediante la escritura de programas de alto nivel.

Guía:

- Escriba los programas en lenguaje C
- Los siguientes operadores de C le pueden resultar útiles:
 - & And bit a bit
 - | Or bit a bit
 - ~ Negación bit a bit
 - ^ Xor bit a bit
 - >> Shift right (n bits)
 - << Shift left (n bits)
- Se puede asumir que los programas ejecutan en un contexto donde:
 - Las variables char se compilan a 8 bits
 - Las variables short se compilan a 16 bits
 - Las variables int se compilan a 32 bits
 - Las variables long se compilan a 64 bits

Preguntas teóricas:

- Explique cómo se pueden utilizar los operadores & y >> para extraer un conjunto de bits dentro de una tira.
- Explique cómo se pueden utilizar las operaciones “/” (división entera) y “%” (módulo) para extraer un conjunto de bits dentro de una tira.
- Explique qué puede cambiar en un programa al definir una variable como 'unsigned char' en lugar de 'char'

Ejercicio 1 ★

Dados los siguientes programa en C, indique el valor binario que almacena cada variable luego de cada asignación:

- short a = 0xA87B;
short b = 0x771C;
short c = a | b;
char d = c >> 4;

- b) `char a = 7;`
`char b = 0xF3;`
`char c = 5;`
`short d = a << 12 | b << 4 | c`
- c) `char a = -3;`
`char b = 0x80;`
`short res = a + b;`

Ejercicio 2 ★

Escribir un algoritmo que permita sumar dos enteros de hasta 2 dígitos, representados como enteros sin signo empaquetado (BCD). El resultado se expresa también en BCD empaquetado. La función tiene el siguiente encabezado:

```
short sumaBCD(char num1, char num2);
```

Ejercicio 3 ★

Escriba funciones que realicen las siguientes conversiones de enteros de 16 bits:

- a) De complemento a 1 a magnitud y signo
b) De complemento a 2 a desplazamiento, con D = 128

Ejercicio 4 ★★

Escribir una función con el siguiente encabezado:

```
int extraerBits(int palabra, int bitAlto, int bitBajo)
```

que dada una palabra devuelve la tira de bits de la palabra que inicia en el número *bitAlto* y termina en *bitBajo*. Por ejemplo, si los parámetros son:

palabra = ...000111001101

bitBajo=2

bitAlto=6, entonces la función debe devolver los bits 6..2 -> 10011 (...00011**10011**01)

Ejercicio 5 ★★

Escribir un programa que calcule la paridad de una palabra de 32 bits.

Ejercicio 6 ★★

Escribir un programa que realice la suma entre dos enteros de 16 bits representados en complemento a 2 y devuelva una palabra de 19 bits con el siguiente formato:

bit 0: bandera Z

bit 1: bandera V

bit 2: bandera C

bits 18..3: resultado de la suma (16 bits)

Ejercicio 7 ★★

Escribir una función con el siguiente encabezado:

short rotar(short numero, int cantidad, char sentido);

Donde *cantidad* indica la cantidad de rotaciones de 1 bit a realizar sobre número, y *sentido* indica el sentido de la rotación (0 = hacia la derecha, otro valor = hacia la izquierda).

Ejercicio 8 ★★

Escriba una función que reciba una tira de 16 bits, representando un número en punto flotante de precisión media (1 bit signo, 5 bits exponente, 10 bits parte fraccional), y devuelva el mismo número representado en punto flotante de precisión simple (1 bit signo, 8 bits exponente, 23 parte fraccional). La función tiene el siguiente encabezado:

int mediaASimple(short num);

Ejercicio 9 ★★★

Sin utilizar variables float, escriba una función que multiplique dos números de 32 bits, que representan números normalizados en punto flotante de simple precisión. La función tiene el siguiente encabezado:

int multiplicarPuntoFlotante(int num1, int num2);