Representación interna de datos en programas

Arquitectura de Computadoras - Práctico 2

Representación Punto fijo

para representar números fraccionarios

Se utiliza una cantidad fija para representar al número, asignando un número fijo de bits para la parte entera y el resto para la parte fraccional.

La representación se obtiene representando en Complemento a 2 el número resultante de multiplicar N por 2^f, siendo f el número de bits dedicados a la parte fraccional.

Tipo fraccionario IEEE 754 de punto flotante

Los números se pueden expresar como $(-1)^s \cdot 1,F \cdot 2^e$

Características:

- Un bit para el signo (s): 0 positivo, 1 negativo
- Mantisa (M) normalizada y desnormalizada
- Base dos para el exponente (e)
- Exponente en desplazamiento

Se definen números normalizados: $(-1)^s \cdot 1,F \cdot 2^e$

y desnormalizados: $(-1)^s \cdot 0,F \cdot 2^{-desp+1}$

Tipo fraccionario IEEE 754 de punto flotante

Recordar

	S (bits)	E (bits)	F (bits)	Total (bytes)
Simple Precisión	1	8	23	4
Doble Precisión	1	11	52	8
Precisión Extendida	1	15	64	10

	E (Exponente)	F (Fracción)
Normalizados	0000….0 < Exp < 1111….1	Cualquier combinación
Desnormalizados	0000……..0	≠ 0
Cero	0000……..0	0
Infinito	1111……..1	0
Not a Number	1111……..1	≠ 0

Tipos de datos enteros

char - 8 bits

short - 16 bits

int - 32 bits

long - 64 bits

- Enteros con signo
- Complemento a 2
- (bool no está permitido en el curso. En los tipos enteros, cualquier valor positivo evalúa a *true*)

Asignaciones

¿Qué valor binario se almacena en las siguientes asignaciones?

```
short num1 =  35;

int num2 = 65536;

char num = 97;

char letra = 'a';
```

Asignaciones (2)

Los resultados se ajustan al tamaño de la variable:

- Si sobran bits, se almacenan los N bits menos significativos:
  - char var8bits = 253 + 61; // 101000100 → 68d
- Si faltan bits, se agregan bits según el signo del resultado (extensión de signo)
  - char var8bits = -3; // 11111101
  - short var16bits = var8bits // 11111111 11111101

Tipos entero sin signo

unsigned char
unsigned short
unsigned int
unsigned long

Almacenan enteros sin signo.
Modifica las operaciones y operadores que exigen interpretación de los datos.

Ejemplo

```
char num1 = 128;
char num2 = 120;
char val = num1 > num2? 100 : 120;
```

¿Qué guarda *val*?
¿Qué ocurriría si las variables fueran unsigned?

Punto Flotante

Tipo float → Se almacena utilizando IEEE 754 de simple precisión

Tipo double → Se almacena utilizando IEEE 754 de precisión doble (64 bits)

En general no se utilizan en el curso, aunque en este práctico aparecen ejercicios que requieren interpretar variables *int* como si fueran *float* (ejs 8 y 9)

Operaciones (1)

Entre variables enteras, el operador menos (-) realiza el complemento a dos.

```
char num = 18;     // 0001 0010
char neg = -num;   // 1110 1110 → -18 en compl a dos
```

¿Qué sucede si la variable *neg* es *unsigned* ?

Las operaciones de suma y resta se implementan con el algoritmo clásico de complemento a 2.

Operaciones (2)

Importante! Recordar que la división es ENTERA

```
short dividendo = 4;
short divisor = 5;
short res = dividendo / divisor;    // res = 0 (!)
```

¿Qué pasa si la variable *res* es float?

```
short dividendo = 4;
short divisor = 5;
float res = dividendo / divisor;    // res = ??
```

Coerción implícita (Promoción de tipos)

- Al operar con variables de diferente precisión, los operandos de menor precisión son promovidos a tipos de datos de mayor precisión.

  Ej: `char num1; short num2; short  res = num1 + num2;`

- Realizada automáticamente por el compilador

char → short → int → long
float → double

Operadores binarios

& → And bit a bit
| → Or bit a bit
^ → Xor bit a bit
~ → Negación bit a bit

>> → Shift hacia la derecha
<< → Shift hacia la izquierda

Ejercicio 1

Indicar los valores binarios de las variables luego de cada asignación

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

```
char a = -3;
char b = 0x80;
short res = a + b;
```

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)

Previo a empezar: *¿Cuál debería ser la firma de la función?*