Manipulación de Bits

B.S. Rodolfo Mercado Gonzales

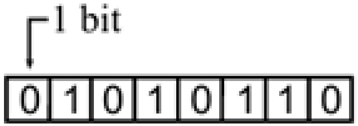
Bit

- Es un dígito binario, puede tomar dos valores 0 (apagado) o 1 (prendido).
- Unidad mínima de información.
- La información procesada en una computadora es codificada en bits.

$$1\ byte = 8\ bits$$

Representación computacional de enteros

- Los enteros son representados como una secuencia de bits.



- De acuerdo al tipo de dato se usa cierta cantidad fija de bits.

short	16 bits (2 bytes)
int	32 bits (4 bytes)
long long	64 bits (8 bytes)

Enteros sin signo

- Se representan en base 2, usando todos los bits disponibles.

Usando 3 bits podemos representar

los enteros en el rango $[\mathbf{0}, \mathbf{2^3} - \mathbf{1}]$

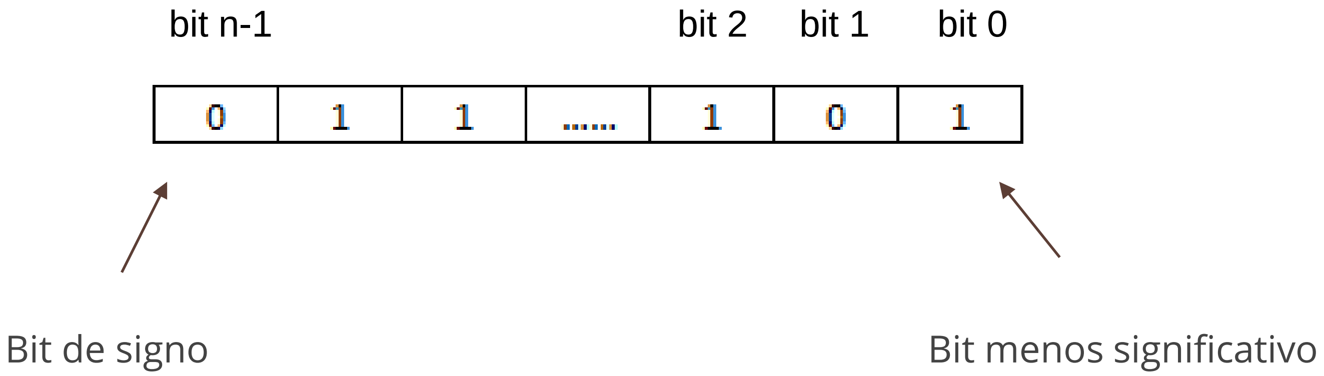
Decimal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Enteros sin signo

- Con $\boldsymbol{n}$ bits podemos representar enteros sin signo en el rango $[\mathbf{0}, \mathbf{2}^{\boldsymbol{n}} - \mathbf{1}]$.

unsigned short	$[0, 2^{16} - 1]$
unsigned int	$[0, 2^{32} - 1]$
unsigned long long	$[0, 2^{64} - 1]$

Complemento a dos



Es la representación que se usa actualmente para entero con signo

Complemento a dos

- **Número positivo o cero**

El bit de signo es igual a 0 y los $n - 1$ bits restantes se completan con la representación binaria del número.

Fijando 3 bits tenemos **000** (0) , **001** (1) , **010** (2) y **011** (3).

Complemento a dos

- **Número negativo**

- El bit de signo es igual a 1 y los $n - 1$ bits restantes se completan con la representación binaria de $2^{n-1} - abs(x)$.

- La misma representación se puede obtener como $\sim\boldsymbol{abs(x) + 1}$ (ver operadores bitwise).

Fijando 3 bits tenemos **100** (-4) , **101** (-3) , **110** (-2) y **111** (-1).

Complemento a dos

Con $\boldsymbol{n}$ bits podemos representar enteros con signo en el rango $[-\mathbf{2}^{\boldsymbol{n}-\mathbf{1}}, \mathbf{2}^{\boldsymbol{n}-\mathbf{1}} - \mathbf{1}]$

short	$[-\mathbf{2}^{15}, 2^{15} - 1]$
int	$[-\mathbf{2}^{31}, 2^{31} - 1]$
long long	$[-\mathbf{2}^{63}, 2^{63} - 1]$

Operadores Bitwise

- ❑ Son similares en tabla de verdad a los operadores binarios (&&, | | , !)
- ❑ Realizan operaciones bit a bit.
- ❑ Se realizan en tiempo constante.

Operadores Bitwise

a	b	~ a (not)	a & b (and)	a | b (or)	a ^ b (xor)
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Operadores Bitwise

Sea $A = 1010 \;\; y \;\; B = 1100$

- $A \,\&\, B \; = 1000$
- $A \mid B \; = 1110$
- $A \wedge B \; = 0110$
- $\sim A \quad = 0101$

Notar que:

A ^ A = 0
A & A = A
A | A = A

Operadores Bitwise

Shift a la derecha $(\boldsymbol{x \gg i})$

Todos los bits de **x** corren $\boldsymbol{i}$ posiciones a la derecha y los nuevos bits son llenados con el bit del signo.

Es equivalente al piso (floor) de la división de $\boldsymbol{x}$ entre $\mathbf{2}^{\boldsymbol{i}}$.

$$\boldsymbol{Sea\ x = 23} \qquad \boldsymbol{x \gg 1 \quad = \quad 00001011 \quad = 11}$$

7	6	5	4	3	2	1	0
0	0	0	1	0	1	1	1

Operadores Bitwise

Shift a la izquierda $(\boldsymbol{x} \ll \boldsymbol{i})$

Todos los bits de **x** corren $\boldsymbol{i}$ posiciones a la izquierda y los nuevos bits son llenados con ceros.

Es equivalente al producto de $\boldsymbol{x}$ por $\mathbf{2}^{\boldsymbol{i}}$.

$$\boldsymbol{Sea\ x = 23} \qquad\qquad \boldsymbol{x \ll 1 \quad = 00101110 \quad = \quad 46}$$

7	6	5	4	3	2	1	0
0	0	0	1	0	1	1	1

Máscara de Bits

- Podemos usar un entero para representar subconjuntos de un conjunto de hasta 32 elementos (o 64 si usamos long long).
- El i-ésimo bit del entero (máscara) es 1 si el i-ésimo elemento del conjunto está presente y será 0 si está ausente.

Máscara de Bits

Supongamos que tenemos el conjunto $\{1, 3, 8\}$, entonces:

Entero	Máscara	Subconjunto
0	000	{ }
1	001	{ 1 }
2	010	{ 3 }
3	011	{ 1, 3 }
4	100	{ 8 }
5	101	{ 1, 8 }
6	110	{ 3, 8 }
7	111	{ 1, 3, 8 }

Tareas Frecuentes

- Unión de conjuntos : $A \mid B$
- Intersección de conjuntos : $A \,\&\, B$
- Diferencia de conjuntos : $A \,\&\, \sim B$
- Obtener bit de posición i : $(\text{mask} \gg \text{i}) \,\&\, 1$
- Encender el bit de posición i : $\text{mask} = \text{mask} \mid (1 \ll \text{i})$
- Apagar el bit de posición i : $\text{mask} = \text{mask} \,\&\, (\sim(1 \ll i))$
- Cambiar estado del bit i : $\text{mask} = \text{mask} \,\hat{}\, (1 \ll \text{i})$

Tareas Frecuentes

- **Obtener último bit encendido de un entero**

Para un $x = 14$, tenemos:

bit	1	1	**1**	0
posición	3	2	1	0

- El último bit encendido nos lo da la operación: $\boldsymbol{x}\ \&\ -\boldsymbol{x}$
- **(-)** es un operador unario que genera el negativo de un número.
- Como los números se guardan en complemento a 2, entonces: $-\boldsymbol{x} = \sim\boldsymbol{x} + \mathbf{1}$

Tareas Frecuentes

Demostración

Sea $x = \overline{a10\ldots0}$

$$-x = \overline{\sim(a10\ldots0)} + 1, \qquad -x = \overline{(\sim a)01\ldots1} + 1, \qquad -x = \overline{(\sim a)10\ldots0}$$

Finalmente,

$$x \,\&\, -x = \overline{a10\ldots0} \;\&\; \overline{(\sim a)10\ldots0} = \mathbf{0\ldots010\ldots0}$$

Tareas Frecuentes

- **Obtener el número de bits encendidos rápidamente**

Podemos obtenerlo hallando el último bit encendido e ir restándoselo al número, repitiendo esto mientras el número sea diferente de 0.

Otra forma es usando las funciones :

- __builtin_popcount
- __builtin_popcountll

Tareas Frecuentes

- **Saber si un número es potencia de 2**

Quitamos el último bit encendido y lo que queda deber ser igual a 0

$$Si \quad x - (x \,\&\, -x) = 0 \,, \qquad es\ potencia\ de\ 2$$

Problemas

HackerRank – Lonely Integer

HackerEarth – Aaryan, Subsequences And Great XOR

HackerEarth – MISTERY

HackerRank – Sum vs Xor

Referencias

- Hackerearth, Bit Manipulation
- Topcoder, A Bit of Fun: Fun with Bits
- Halim, Steven and Halim, Felix. Competitive Programming 3

¡ Good luck and have fun !