Manejo de Bits en Lenguaje C

Arquitectura del Computador LCC-FCEIA

Septiembre de 2020

Operadores de bits en C

Operador	Acción
&	Operación AND bit a bit
|	Operación OR bit a bit
^	Operación XOR bit a bit
~	Operación NOT bit a bit
>>	Desplazamiento a la derecha
<<	Desplazamiento a la izquierda

Operadores de bits en C

Importante!

Los operadores de bits pueden ser aplicados a variables de tipo entero ya sea con signo o sin signo (`char`, `short`, `int`, `long`, etc.) y NO pueden ser aplicados a variables tipo flotante (`float`, `double`, etc.).

Operador AND

La operación AND esta definida por la siguiente tabla de verdad:

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

Supongamos que tenemos dos variables, a=56 y b=72, entonces la operación AND entre estas dos variables resulta:

$$\begin{array}{cccccccc} & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \& & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

Es decir, a&b=8.

Operador AND

Ejemplo

Dado a=72, se quiere averiguar si el cuarto bit de dicho número es 1 o 0:

$$\begin{array}{cccccccc} & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ \& & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

Por lo tanto, al realizar a&b vemos que efectivamente el cuarto bit del número es 1.

Operador OR

La operación OR esta definida por la siguiente tabla de verdad:

A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1

Supongamos que tenemos dos variables, a=56 y b=72, entonces la operación OR entre estas dos variables resulta:

$$\begin{array}{cccccccc} & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ | & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}$$

Es decir, a|b=120. Notar que el resultado es distinto al obtenido si se utiliza el operador OR lógico (||).

Operador OR

Ejemplo

Dado número a=28, si queremos poner en uno el séptimo bit (b_6) podemos realizar a=a|b, donde b=01000000b:

$$\begin{array}{cccccccc} & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ | & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{array}$$

Podemos observar que el séptimo bit de a ahora es uno.

Operación XOR

La operación XOR está definida por la siguiente tabla de verdad:

A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

Supongamos que tenemos dos variables, a=56 y b=72, entonces la operación XOR entre estas dos variables resulta:

$$\begin{array}{cccccccc} & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \hat{} & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}$$

Es decir, a^b=112.

Propiedades del operador XOR

- Conmutativa: `A^B = B^A`
- Asociativa: `A^(B^C) = (A^B)^C`
- `A^A = 0`
- `A^0 = A`

A partir de las propiedades anteriores resulta:

- `(B^A)^A = B^0 = B`

Operador XOR

Ejemplo

La cadena de caracteres “Hola” se puede representar utilizando código ASCII como 01001000 01101111 01101100 01100001.

Esta cadena puede ser cifrada con la clave 11110011:

$$
\begin{array}{ll}
 & 01001000\ 01101111\ 01101100\ 01100001 \\
\hat{} & 11110011\ 11110011\ 11110011\ 11110011 \\
\hline
 & 10111011\ 10011100\ 10011111\ 10010010
\end{array}
$$

Aplicando el operador XOR con la misma clave:

$$
\begin{array}{ll}
 & 10111011\ 10011100\ 10011111\ 10010010 \\
\hat{} & 11110011\ 11110011\ 11110011\ 11110011 \\
\hline
 & 01001000\ 01101111\ 01101100\ 01100001
\end{array}
$$

Operador complemento a uno

El operador binario ~ se conoce como operador complemento a uno o también como operador NOT. Es un operador unario, es decir solo necesita un operando.

A	~A
0	1
1	0

Si a=56, resulta ~a=-57 dado que la secuencia de bits que representa a la variable a es $00111000b$ y la secuencia que resulta de aplicarle el operador ~ resulta $11000111b$.

Operador complemento a uno

Ejemplo

La expresión `x=x&~0xff` pone los últimos 8 bits de `x` en cero.

Notar que esto es independiente de la longitud del tipo de dato a diferencia de la expresión `x=x&0xffffff00` que asume que `x` tiene 32 bits.

Operadores desplazamiento

- Se pueden realizar corrimientos de N bits, con $N \in \mathbb{N}$, es decir $N = 1, 2, \ldots$.
- Los desplazamientos de bits pueden ser hacia la derecha o la izquierda.
- Cada vez que se hace un desplazamiento se completa con ceros en el otro lado (no se trata de una rotación).

Ejemplo

Si a=56 ($56_{10} = 00111000b$) y le realizamos un corrimiento de un bit hacia la derecha: a>>1. El resultado es $00011100b = 28_{10}$.

Notar que se ha dividido al número a por dos!

Operadores desplazamiento

Desplazando el número n bits hacia la derecha se realiza la división entera del número:

$$a>>n = a/2^n$$

Análogamente, el desplazamiento a la izquierda es equivalente a multiplicar por potencias de dos:

$$a<<n = a \times 2^n \tag{1}$$

La ventaja de los desplazamientos es que son menos costosos que hacer las operaciones de multiplicación y división.

Operadores desplazamiento

Importante: Hay que tener cuidado al realizar desplazamientos hacia la izquierda dado que se obtener resultados erróneos debido al *overflow*.

También hay que tener precauciones la desplazar hacia la derecha:

- En los desplazamientos hacia la derecha de valores sin signo siempre se completan los bits vacíos con ceros.
- En los desplazamientos hacia la derecha de valores con signo se completa de acuerdo al bit de signo ("*arithmetic shift*") en algunas máquinas y con ceros ("*logical shift*") en otras.

Operadores desplazamiento

Ejemplo

Determinar si el bit b_3 de un número en binario es 0 o 1:

```
(a>>3)&1
```

Si el resultado es 1, indica que efectivamente el bit b_3 es 1.