Bitwise Operation (1A)

Young Won Lim
12/15/16

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

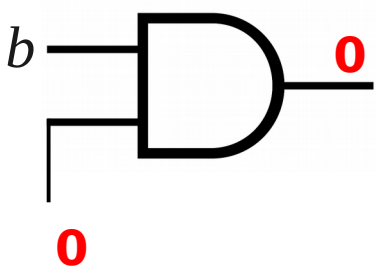
Young Won Lim
12/15/16

Bitwise Operators

Logical AND	&&
Logical OR	||
Logical XOR	n/a
Logical NOT	!

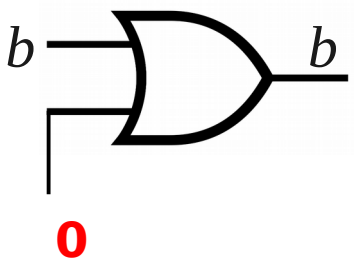
bitwise AND	&
bitwise OR	|
bitwise XOR	^
bitwise NOT	~

Logic Gates and Bit Operators

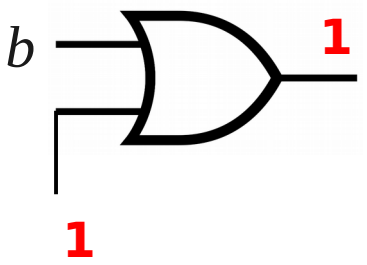


b & 0 = 0

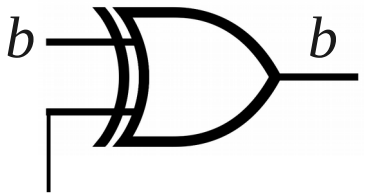
b & 1 = b



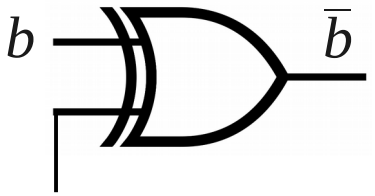
b | 0 = b



b | 1 = 1



b ^ 0 = b



b ^ 1 = $\overline{\text{b}}$

Testing Bits

if (x & 64) != 0)

if (x & 0x0040)

if (x & (1 << 6))

testing bit 6 of x

0x0040 : 64 in hex
1 << 6 : 2^6 = 64

Setting & Clearing Bits

Setting bit 7 of x

x = x | (1 << 7)

x |= (1 << 7)

Clearing bit 7 of x

x = x & ~(1 << 7)

x &= ~(1 << 7)

Inverting bit 7 of x

x = x ^ (1 << 7)

x ^= (1 << 7)

0x0080 : 64 in hex
1 << 7 : 2^7 = 128

Max Shifts for 32-bit and 64-bit numbers

32-bit shift

1 << 30	signed 2^{30}
1U << 31	unsigned 2^{31}

64-bit shift

1L << 62	signed 2^{62}
1UL << 63	unsigned 2^{63}

32-bit shift

1 << 31	(overflow)
1U << 32	(overflow)

64-bit shift

1L << 63	(overflow)
1UL << 64	(overflow)

Max Numbers for 32-bit and 64-bit numbers

32-bit shift

((1U << 31) – 1) + (1U << 31) $2^{32} - 1$

64-bit shift

((1UL << 63) – 1) + (1UL << 63) $2^{63} - 1$

Difference?
1 << 32
1 << 32L
1 << 16
1 << 16L

References

[1] Essential C, Nick Parlante
[2] Efficient C Programming, Mark A. Weiss
[3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
[4] C Language Express, I. K. Chun
[5] “A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux“
http://cseweb.ucsd.edu/~ricko/CSE131/teensyELF.htm
[6] “Fundamentals of Embedded Software ...”, D.L. Lewis