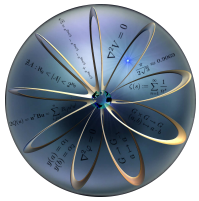

planetmath.org

Math for the people, by the people.

computer representation of integers

Canonical name	ComputerRepresentationOfIntegers
Date of creation	2013-03-22 16:55:55
Last modified on	2013-03-22 16:55:55
Owner	rm50 (10146)
Last modified by	rm50 (10146)
Numerical id	5
Author	rm50 (10146)
Entry type	Topic
Classification	msc 68P20
Classification	msc 11A63

Computer systems that use a hardware binary representation for integral values typically have one of two choices available to them for the representation of negative integers: one's complement or two's complement.

Generally the high-order bit of a binary representation is used as a sign bit. If the value is 1, then the number is negative, while if the value is 0, the number is nonnegative.

In a computer that uses one's complement notation, the representation of a negative number is the one's complement of the representation of a positive number. In other words, the negation of an integer can be accomplished by exclusive or'ing its value with a word consisting of all 1's. So, in one's complement, assuming a 4-bit word for compactness,

$$\begin{aligned} 1 &= 0001 \\ 4 &= 0100 \\ -1 &= 1110 \\ -2 &= 1101 \end{aligned}$$

Two's complement, as its name implies, represents negative integers by taking their complement with respect to the next higher power of 2. Thus, in a 4-bit world, the negative of a value is found by subtracting it from 2^5 (as a bit string). Thus, for example, $1 = 0001$, so $-1 = 10000 - 0001 = 1111$.

An unfortunate result of using one's complement notation is that zero has two distinct representations:

$$0 = 0000 = 1111$$

Two's complement fixes this, since $-0 = 10000 - 0000 = 10000$, which gets truncated in 4 bits to 0000. However, it introduces an asymmetry in the representational range. While a 4-bit one's complement number can represent any integer from -7 to $+7$, a 4-bit two's complement number can represent integers from -8 to $+7$, -8 being represented by the value 1000.

In addition, two's complement is in some sense more natural than one's complement since if you regard the representation as an unsigned number, its value modulo $2^{\text{word size}}$ is in fact the value it represents. In other words, for example, consider again a 4-bit word size. Then the representation of -2 in two's complement is 1110. Thinking of 1110 as an unsigned number, its value is 14, which is in fact $-2 \pmod{2^4}$.

This fact means that multiplication of two's complement representations is quite easy - one need only multiply the bit patterns together as if they

were unsigned, and consider the rightmost bits of the result (thus taking the result modulo the word size).

Using one's complement notation, no such simple algorithm is possible. Thus, for example, in one's complement one has

$$\begin{aligned} -1 &= 1110 \\ -2 &= 1101 \\ 2 = -1 \cdot -2 &= 0010 \end{aligned}$$

yet mutiplying the unsigned values and truncating results in 0110, or 6.

This inefficiency in required multiplication algorithms, together with the obvious problem of having multiple representations of zero, has led to the almost complete abandonment of one's complement notation today.