IB Computer Science Revision Notes

Two's complement

Computers have difficulties dealing with signed numbers. Why? Because that adds a lot to the complexity of the calculations because it implies having a negative and a positive 0. To overcome this problem, the number line is divided in two parts with 0 in the middle. Left of 0 are the number from the upper half of the number line in descending order while right from 0 are the lower half of the number line in ascending order, like on the picture below.

Signed	Unsigned	Integer 2's Complement
5	5	0000 0101
4	4	0000 0100
3	3	0000 0011
2	2	0000 0010
1	1	0000 0001
0	0	0000 0000
-1	255	1111 1111
-2	254	1111 1110
-3	253	1111 1101
-4	252	1111 1100
-5	251	1111 1011

This way makes dealing with negative numbers easier. For example, if 5 is subtracted from 2, the value of -5 can be added to two, with any carry above the 8th bit discarded:

_00000010
+11111011
_11111101

As you can see these add up to 253, which is the value of -3. By the way, this works in any number system, even in our decimal system.

The problem with this representation is when you try to add values where the result goes out of the borders of the number line. For example, if you would try to add 28 to 127 in our example 8 bit integer, you would correctly add 00011100 to 01111111:

_00011100
+01111111
_10011011

But, 10011011 is not 155, in our number line it corresponds to -100! As you can see, by adding to values that give more than 127, we accidentally have gotten into the -ve values of the number line, because the 8th bit is a sign bit. This is called an overflow.

Created by Matyas Mehn — *Matyas Mehn [mailto:matyas.mehn@gmail.com] 2014/03/28 11:34*