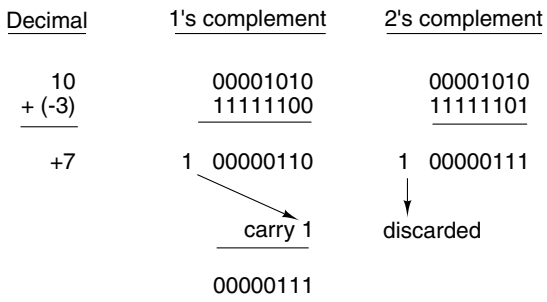
Detecting Overflow in Binary Integer Arithmetic

Two binary numbers can be added, starting at the rightmost bit, using the familiar biwise-addition rules:

Addend	0	0	1	1
Augend	+0	+1	+0	+1
Sum	0	1	1	0
Carry	0	0	0	1

Note, in the above diagram, the technical terminology of *addend* (i.e. the object that is being added to) and *augend* (i.e. the quantity being added). If a carry is generated, it is carried one place to the left just as in decimal arithmetic. In one's complement arithmetic a carry generated by the addition of the leftmost bits is added to the rightmost bit. This process is called an *end-around carry* In two's complement arithmetic a carry generated by the addition of the leftmost bits is merely thrown away. An example is shown in the following diagram:



If the addend and the augend are of opposite signs an overflow error cannot occur. If they are of the same sign and the result is of the opposite sign then an overflow error has occurred and the result is wrong. In both one's and two's complement arithmetic, overflow occurs if, and only if, the carry *into* the sign bit differs from the carry *out of* the sign bit. Most computers preserve the carry out of the sign bit (in the C flag of the CPSR in ARM) but the carry into the sign bit is not visible from the answer. For this reason a special overflow bit is also provided. This overflow bit is yet another of the flags in the CPSR of the ARM chip.