MITOCW | MIT6_004S17_01-02-12-03_300k

The nice feature of two's complement representation is that it allows you to do both addition and subtraction as addition problems, by simply turning the subtraction problem A-B into an addition problem of A + (-B).

Since we now know how to easily negate values in two's complement, this is pretty straight forward.

For example, let's perform 15 - 18 using 6-bit two's complement representation.

15-18 = 15 + (-18) 15 = 001111 18 = 010010 To convert 18 to -18, we flip all the bits and add one.

This results in 101110 which is = -18.

We can now take our two's complement representation of our two numbers and add them together.

Whenever we add 1 + 1 in binary that produces a 0 plus a carry of 1.

So our final sum is 111101.

To see what this number is equal to, we again flip the bits and add one.

Flipping the bits, we get 000010, and adding one to that we get 000011 which equals 3.

Which means that 111101 = -3 which is the result we expected to get when performing 15-18.

Let's try another example: 27 - 6 = 27 + (-6) 27 is equal to 16 + 8 + 2 + 1 which is equal to 24 + 23 + 21 + 20.

So in two's complement representation this number is 011011.

6 is equal to 4 + 2 which is equal to 22 + 21, so its 6-bit binary representation is 000110.

To find the representation of negative six, we flip all the bits and add one resulting in 111010.

When we add these two numbers together, remembering that in binary 1 + 1 produces a 0 plus a carry of 1, we find that our result ends up having 7 bits.

This most significant bit is dropped because we are working with 6-bit representation in this case.

So our final result is 010101 which is equal to 24 + 22 + 20 which equal 16 + 4 + 1 or 21.

If you try adding two numbers whose result is beyond the range of numbers that can be represented using your N-bit representation, then that means that overflow occurred.

Overflow can be detected by looking at the sign of the two numbers being added and the sign of the resulting number.

If you try adding two positive numbers (where the most significant bit is 0) and your result ends up being negative (your most significant bit is a 1), then overflow occurred.

Similarly if you try adding two negative numbers whose most significant bit is one, and you end up with a result that is positive (where the most significant bit is 0) then once again overflow occurred.

Overflow cannot occur when you add a positive and negative number that are within range.

Let's take a look at an example of this.

Suppose we try adding 31 + 12 using 6-bit two's complement.

This results in 101011 which has its most significant bit equal to 1 even though we were adding two positive numbers.

This means that overflow occurred.

Overflow occurred because the result of adding these two numbers is 43 which is larger than 2 to the 5th minus 1, or 31, which is the largest positive number that can be represented using 6-bits two's complement.