Sign-magnitude representation

- The sign of a number is stored as an additional piece of information.
- This corresponds to our usual mathematical notation
    - +3, -102
- One of the bits, usually the left-most, is used for the sign. The remainder of the sequence represents the absolute value of the number
    - We use 0 for + and 1 for -
- In a 4-bit representation
    - +3 is represented as **0**011
    - -3 is represented as **1**011
- The issue with this approach is arithmetic:
    - Adding two positive numbers works
    - Adding two negative numbers works
    - Adding two negative **does not** work
- Another issue with this approach is that there is two 0's which is wasting a representation

Excess representation

- Excess representation starts at -2^{n-1} and goes up to $2^{n-1}-1$
- If n = 8 we have
    - 00000000 represents -128
    - 11111111 represents 127
- If we write the bit strings in alphanumeric order, 0 will be just below the middle, with the positive numbers below and the negatives above. The bias is 0
- Its possible to do arithmetic with excess, but its not natural:
    - In order to do x + y you have to do:
        - $x + 2^{n-1} + y + 2^{n-1}$
        - subtract 2^{n-1}
        - $x + y + 2^{n-1}$ -> answer in excess representation
        - decode
        - x + y -> answer
    - (double check)
- You can also change the bias when using excess representation