

**2’s complement representation:** {<https://en.wikipedia.org/wiki/Two%27s_complement>}

Advantages: {<http://sandbox.mc.edu/~bennet/cs110/textbook/module3_2.html>}

* Fundamental arithmetic operations of addition, subtraction, multiplication are identical to those for unsigned binary numbers
  + Input are represented in the same number of bits as the output, any overflow beyond those bits can be discarded from the result
  + Can be used in natural addition and subtraction arithmetic without any need to change the bits (providing no overflow occurs)
  + Allows negative and positive numbers to be added or subtracted with each other without any special logic (normal way)
* Simpler to implement (higher-precision arithmetic)
* No representation for negative zero
  + -0 is just a separate way to represent 0 that would not equal to 0 with the normal bit-by-bit method of comparing numbers)
* Easy to get 2’s complement = just add 1 bit to 1’s complement
* Identical to unsigned binary numbers
* Same as binary representation of positive numbers; for negative numbers have to flip from 0 to 1 and 1 to 0 from binary representation of negative numbers to get 1’s complement and add 1 to get 2’s complement of a negative number
  + 1 extra bit to represent negative numbers

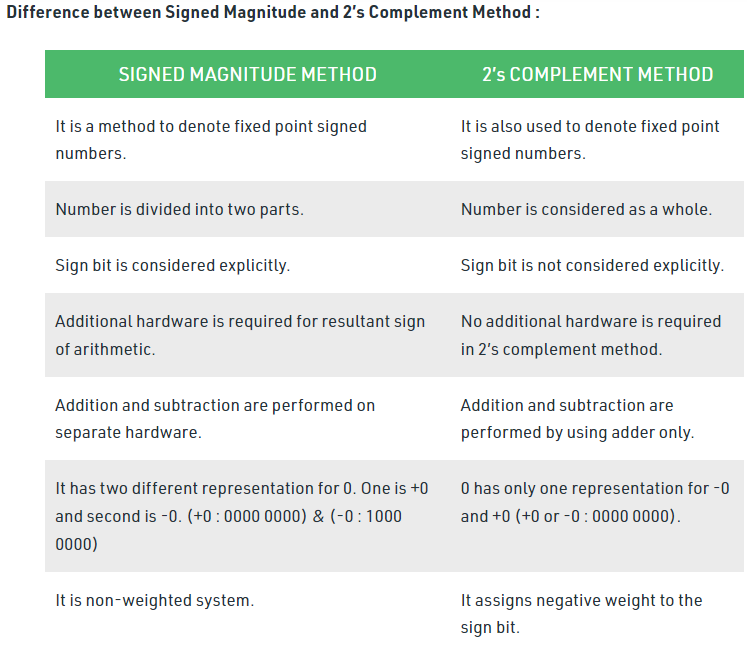
Disadvantages: {<https://stackoverflow.com/questions/18952272/twos-complement-disadvantage>}

* If one tries to negate the lowest representable value, you’ll get an overflow
  + Doesn’t happen with 1’s complement & sign-and-magnitude

RESOURCES =

<https://quick-adviser.com/what-are-the-disadvantages-of-2s-complement/#What_are_the_disadvantages_of_2s_complement>

RESOURCES = <https://www.geeksforgeeks.org/difference-between-signed-magnitude-and-2s-complement/>

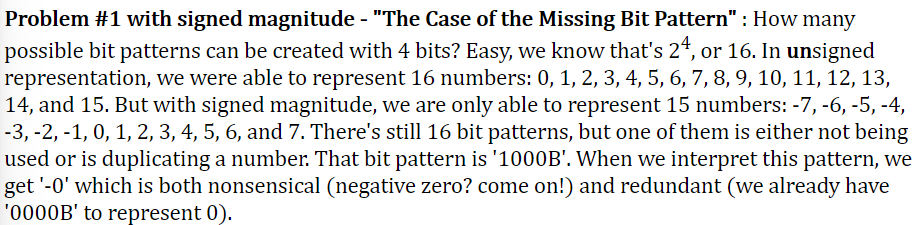
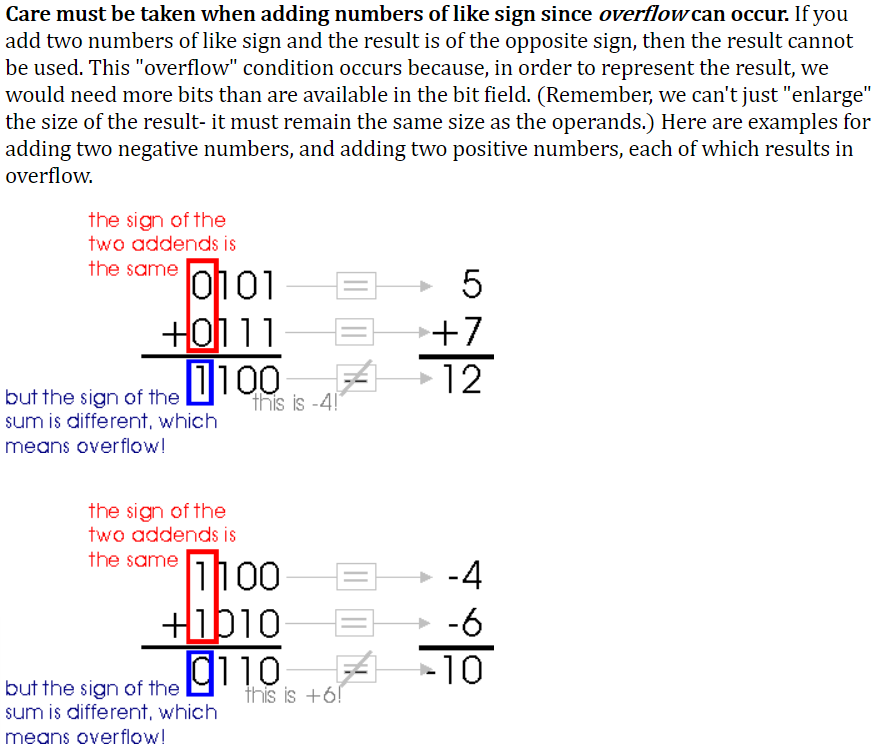
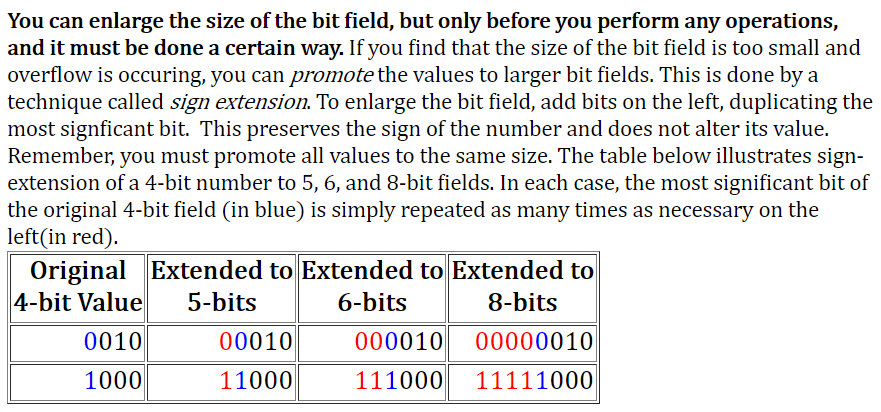
****

**Sign-and-magnitude representation:**

Advantages:

* Won’t get an overflow when one tries to negate the lowest representable value
* Compared to unsigned numbers, one can work with negative numbers but the range of positive values that can be worked with will be halved
* Simplest and most common methods to represent positive and negative numbers {<https://www.electronics-tutorials.ws/binary/signed-binary-numbers.html>}
  + Can determine whether a number is negative or non-negative (positive) simply by testing the most significant bit
    - 1 = negative; 0 = positive
    - Left-most bit
    - Remaining bits = magnitude of the number

Disadvantages:

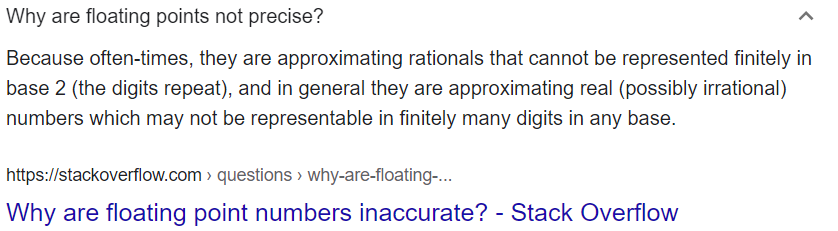
* One of the bit patterns is wasted to represent if the number is a positive or negative number so addition can’t be done is the normal way (regular binary addition)
  + Once one have chosen a bit-field width (eg. 28-bits), one must stay with it, if the number has 25-bits, either 0 or 1 is use to fill the blank bits of the bit-field width or if the bit-field width is just nice to fill in the bit-field width then the left-most bit is sacrificed to represent the number which is either positive or negative
* Probability of missing bit pattern
  + 
* 
* 

**Scientific notation *(floating point)* representation:**

Advantages: {<https://www.indeed.com/career-advice/career-development/how-to-write-in-scientific-notation#:~:text=Benefits%20of%20scientific%20notation&text=Scientific%20notation%20ensures%20accuracy%20and,working%20with%20such%20large%20numbers>.} {<https://www.oreilly.com/library/view/c-primer-plus/9780132781145/ch03lev2sec13.html#:~:text=Floating%2Dpoint%20numbers%20have%20two,much%20greater%20range%20of%20values>.}

* (SN) Ensures accuracy and reduce the possibility of error when using very small or very large numbers
  + Manageable & easier to keep track of
* (SN) Easier to interpret & deal with very large or very small numbers
  + During arithmetic problem-solving
  + Easily read without distraction of non-significant values (the zeroooossss)
  + Easy comparison between numbers
  + Convenient
* Represent values between integers
* Scaling factors
  + Can represent a much greater value = support wider range of values
    - Very small or very large

Disadvantages: {<https://wiki.sei.cmu.edu/confluence/display/c/FLP00-C.+Understand+the+limitations+of+floating-point+numbers>}

* Lose precision
  + Do not represent all of the same values
  + Not binary compatible
  + Prone to errors associated with roundings
  + 
* Only store real integers

