Floating Point Representation

Normalization

- To represent the number (+1000111.0101):
  1. Store the sign.
  2. All of the bits.
  3. The position of the decimal point.

- This is possible but it makes operations on the numbers difficult. So, we need a standard representation for floating-point numbers
- Normalization: moving the decimal point. So, there is only one 1 to the left of the decimal point ⇒ 1.xxxxxxx

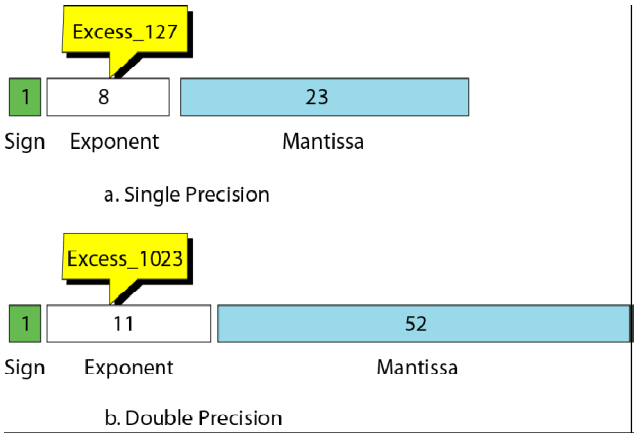
- How to normalize?
  - To indicate the original value of the number, multiply the number by 2^e.
  - Where e is the ***exponent*** that indicate the number of bits that the decimal point moved.
    - Positive ⇒ left moved
    - Negative ⇒ right moved

In General: $\pm 1.xxxxxx \times 2^{\pm e}$

- After the number is normalized we store only three pieces:
  1. ***Sign***: the sign of the number can be stored using 1 bit (0 or 1)
  2. ***Exponent***: power of 2 (+v e or –v e)
  3. ***Mantissa***: the binary number to the right of the decimal point.

IEEE standards for floating-point representation

- IEEE= Institute of Electrical and Electronics Engineers
- Defined standers for storing floating-point numbers:
  1. Single precision format = use 23 bits
  2. Double precision format = use 52 bits



Single precision representation:

1. Store the sign (+ve ⇒ 0 , -ve ⇒ 1)
2. Store the exponent as Excess-127
    a. 127+(±e)
    b. Convert result of a. to binary.
    c. Store the resulted binary value as exponent.
3. Store the mantissa as unsigned integer.

Ex: Show the representation of the normalized number + 1.01000111001×2^6 in single precision?

Solution:

1. The sign is positive ⇒ 0
2. The Excess_127 representation of the exponent 6
    a. 127+(+6)=133
    b. 133= 10000101
3. Add extra 0s to the right of mantissa to make it 23 bits.

The number in memory is stored as:

0 10000101 01000111001000000000000

Interpretation:

To interpret a floating-point number stored in memory:

1. Use the leftmost bit as the sign.
2. Change the next 8 bits (exponent) to decimal and subtract 127 from it.
3. Add 1 and decimal point to the next 23 bits. You can ignore any extra 0s at the right.
4. Move the decimal point to the correct position using the value of the exponent.
5. Change the whole part to decimal.
6. Change the fraction part to decimal.
7. Combine the whole and the fraction.

Ex: Interpret the following 32-bit floating-point number
1 01111100 11001100000000000000000

1. The leftmost bit is 1, so the sign is negative.
2. The next 8 bits are 01111100 (exponent) the 124 – 127= –3
3. The number after normalization is : - 1.110011×2^{-3}
4. Move the decimal point to the correct position using the value of the exponent:
    -0.001110011
5. Change the whole part to decimal = 0
6. Change the fraction part to decimal (0.001110011)

$$\overset{2^{-1}}{0.}\overset{2^{-2}}{0}\overset{2^{-3}}{0}\overset{2^{-4}}{1}\overset{2^{-5}}{1}\overset{2^{-6}}{1}\overset{2^{-7}}{0}\overset{2^{-8}}{0}\overset{2^{-9}}{1}1 = 2^{-3}\times 2^{-4}\times 2^{-5}\times 2^{-8}\times 2^{-9} = \frac{1}{8}+\frac{1}{16}+\frac{1}{32}+\frac{1}{256}+\frac{1}{512}$$

= 0.2246

7. Combine the whole and the fraction = -0.2246