Floating Point Numbers

- **Representing *decimal* point numbers representation**
  - A **floating point number** can be written as **2 numbers**:
    - An **exponent**, and
    - A **mantissa**
  - **Example:**
    - 314.159 = 0.314159 x 10^3 = 314159 x 10^{-3}

    (Remember that **multiply/dividing by 10** with **decimal number** can be accomplished by **shifting the decimal point one place to right/left**)

  - We **do not** need to record the fact that the **exponent used is 10** (it is **assumed**)

    Then, we can represent the number 314.159 using a **pair of numbers**:

    - 314.159 = **(314.159, 0)** = **(0.314159, 3)** = **(314159, -3)**

    The **first number** is the **mantissa** and the **second** is the **exponent**.

- **Representing *binary* floating point numbers**
  - **Binary floating point numbers** can also be written as an ***exponent*** and a ***mantissa***, but now using **powers of 2** (instead of 10 for ***decimal*** **numbers**):
  - **Example:**
    - 1010.1011 = 0.10101011 x 2^4 = 10101011 x 2^{-4}

    (Similarly, **multiply/dividing by 2** with ***binary*** **number** can be accomplished by **shifting the decimal point one place to right/left**)

  - Again, we **do not need** to record the fact that the **exponent used is 2** (because every number inside the computer is in binary).

    We can represent the number 1010.1011 using a **pair of binary numbers**:

    - 1010.1011 = **(1010.1011, 0)** = **(0.10101011, 00000100 (4))** = **(10101011, 11111100 (-4))**

    The first number is the mantissa and the second is the exponent.

- **The IEEE Standard for floating point representation**
  - IEEE is a standard making organization (Institute of Electric and Electronics Engineers).
  - IEEE has defined a number of floating point number formats (single precision and double precision)
  - The **IEEE Single Precision represention**:
    - Uses 32 bits (4 bytes)
    - **Format:**

```
             S EEEEEEEE MMMMMMMMMMMMMMMMMMMMMMM
     Bit:    0 1      8 9                     31

     S = sign of the mantissa
     M = mantissa
     E = exponent
```

    - The **mantissa** uses the **sign/magnitude representation**:
      - `S` = the **sign** of the mantissa
      - `MMMM...M` = the **magnitude** of the mantissa
    - The **mantissa** is **normalized** so that:
      - **1.0 <= M < 2.0**
      - The **leading digit 1** is **omitted**
    - The **exponent** `EEEEEEEE` uses the ***excess 127* encoding scheme** for **signed numbers** that is ***similar*** to the **2's complement representation**.

      The **excess 127 encoding** for **8 bits**:

```
      Bit pattern         Encodes the value
     -------------------------------------
      00000000            -127
      00000001            -126
      ....
      01111111            0
      10000000            1
      10000001            2
      ....
      11111111            128
```

  - **Example:**

```
     Given the following floating point representation:

         01000000101000000000000000000000
     =   0 10000001 01000000000000000000000
```

```
   Meaning of each bit according to the IEEE format:

       S EEEEEEEE MMMMMMMMMMMMMMMMMMMMMMM
       0 10000001 01000000000000000000000
Bit:   0 1      8 9                     31
```

```
   Sign of mantissa = 0 (positive mantissa)

   Mantissa bits = 01000000000000000000000

   Represented mantissa:   1.01000000000000000000000 (leading 1 was omitted)
                         = 1.01

   Exponent = 10000001

   Value represented by Exponent = 2 (decimal)
```

```
Therefore:

   value represented = 1.01 (binary)  with exponent 2^2
                     = 101  (binary)
                     = 5 (decimal)
```

Hands-on experimention: click here

- The IEEE Double Precision represention:
  - Uses 64 bits, and it is similar to the single precision format

```
S EEEEEEEEEEE FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0 1         11 12                                                63
```

  - The first bit is the sign bit, S of the **mantissa**
  - The next 11 bits are the exponent bits, 'E',
  - The final 52 bits are the mantissa 'M':
  - The mantissa is normalized so that 1.0 <= M < 2.0

- **Explore further....**

  Here is a nice webpage where you can construct floating point representations: **click here**

  When you explore the above webpage, you must know that there are 2 things that are strange in the IEEE float representation

  1. Because the first bit of the mantissa is always 1 (stop, think, why is that so ? because we made it so: 1.0 <= mantissa < 2.0), it is not stored.

     So if the mantissa bits in a single precision representation are 01010101010101010101010, the actual mantissa bits are **1**.01010101010101010101010 (and the mantissa is between 1.0 and 2.0)

  2. The exponent uses a modified 2's complement encoding, called "excess" encoding.

The single precision exponent uses the "excess 127" encoding which uses 01111111 to represent 0:

```
Bit pattern:
    01111100   01111101   01111110   01111111   10000000   10000001   10000010
    ----+----------+----------+----------+----------+----------+----------+-
       -3         -2         -1          0          1          2          3
Value represented:
```