Computer Limitations and Representation

- Recall the Universal Computer
  - There is a limited tape size to perform calculation
- Recall the von Neumann and Harvard architecture
  - There is a predefined width to registers and memory
- Abstract representations with limited sizes for:
  - Natural Numbers & Zero: unsigned char, unsigned int
  - Integers: short, int, long
  - Rational/Real:
    - Fix Point ---
    - Floating Point float, double

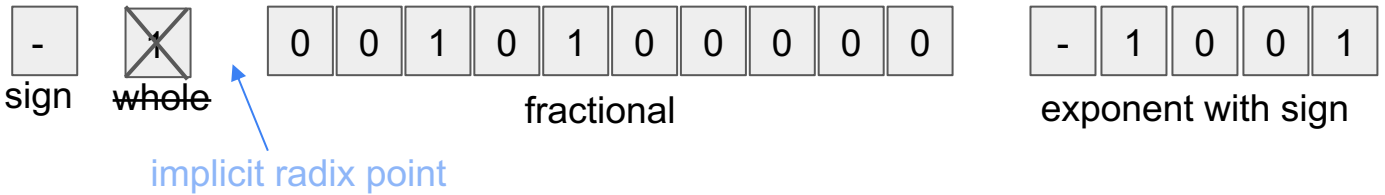
+4.225 x10^ +2
+1.010101 x2^ +101

- An encoding of each will include one or more of the following:

Scientific Notation

$$\begin{array}{rr} & 14.3 \times 10^7 \\ - & 9.2 \times 10^7 \\ \hline & 5.1 \times 10^7 \end{array}$$

- All numbers represented as: m x 10^N
- Simplifies operations on large and small numbers.
  - Distance between sun and earth: 92,000,000 = 9.2 x 10^7
  - Distance between sun and mars: 143,000,000 = 1.43 x 10^8
- Floating point representation
  - a representation of scientific notation
  - introduces the notion of infinity, and NaN (0 / 0 = ?, 0 x infinity = ?)
- Representation of: -1.00101 x 2^{-1001}
  - Assume a size of 16 to hold all necessary information
  - Note the whole part is always "1"



Floating Point Encoding

always 1: so we don't store it

Original number: 2# - 0.000100101

Recall Scientific Notation: - 1.00101 x 2 – 100 // 4

- Components to Encode
  - sign: negative
  - coefficient: “1.00101” and the mantissa: “00101”
  - exponent: - 100
    - Issue: negative exponents
    - Solution: store values with a bias
- Bias:
  - Shift all numbers along the number line by N
  - Typically it is half the range:
    - 3 bits -> 011 == 3
    - 5 bits -> 0 1111 == 15
    - 8 bits -> 0111 1111 == 127
    - 11 bits -> 011 1111 1111 == 1023

Symbol	Encoding
+	0
-	1

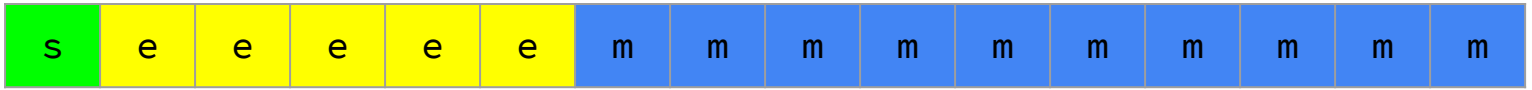
Number		Encoding (Bias: 4)
-4		000
-3		001
-2		010
-1		011
0	000	100
1	001	101
2	010	110
3	011	111

Floating Point Encoding

https://en.wikipedia.org/wiki/Single-precision_floating-point_format

Recall Scientific Notation: -1.00101×2^{-100} // -4

- Formats:
  - binary16 (half): 1 + 5 + 10 = 16, 0 1111 = 15



  - binary32 (single): 1 + 8 + 23 = 32, 0111 1111 = 127

  - binary64 (double): 1 + 11 + 52 = 64, 011 1111 1111 = 1023

Floating Point Encoding

Recall Scientific Notation: $- 1.00101 \times 2^{-100}$ // -4

- Half Precision
  - float16 (half): `1 + 5 + 10 = 16,` `0 1111 = 15`

- Components
  - `sign: 1`
  - `mantissa: 0010100 ; fill in least significant bits with zero (0)`
  - `expon: -4 + 15 = 11 → 1011`

Floating Point Encoding

Recall Scientific Notation: - 1.00101 x 2^{-100} // -4

- Single Precision
  - float32 (single): 1 + 8 + 23 = 32, 0111 1111 = 127

- Components
  - sign: 1
  - mantissa: 001010 ; fill in least significant bits with zero (0)
  - expon: -4 + 127 = 123 → 0111 1011