# Fixed and Floating-point Numbers

Eric McCreath

# Fractional binary numbers

- Remember how the meaning of the digits in a binary number is defined:

$$...d_2 d_1 d_0 . d_{-1} d_{-2} ... = \sum_i d_i \times 2^i$$

Note the binary radix point

- For example, the binary number:

$$10.01_{(2)}$$

means

$$1 * 2^1 + 0 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2} = 2.25_{(10)}$$

# Binary

- Converting a fractional number (represented as a decimal) to a fractional binary number works by repeated multiplication by 2. This effectively shifts the digits of the binary number past the unit digit. As the digits pass the unit digit they can be recorded.

- For e.g. to convert 0.6 to a fractional binary number:

$$
\begin{aligned}
0.6 * 2 &= 1.2 \\
0.2 * 2 &= 0.4 \\
0.4 * 2 &= 0.8 \\
0.8 * 2 &= 1.6 \quad \Rightarrow 0.10011001100110011..._{(2)} \\
0.6 * 2 &= 1.2 \\
0.2 * 2 &= 0.4 \\
&\vdots
\end{aligned}
$$

# Fixed-point Numbers

- Fixed-point number representation provides a way for computers to store fractional numbers
  - A standard signed/unsigned integer is stored and this is scaled by a fixed factor determined by the type
  - This is like shifting the radix point a fixed number of places to the left
- For e.g. consider an 8 bit unsigned integer with a scaling factor of 1/8, then:

$$00110101 \rightarrow 110.101 \rightarrow 6.625_{(10)}$$

- Fixed-point representation is simpler than floating-point for performing calculations

# Floating-point Numbers

- Floating-point numbers provide a way of representing real numbers with a wide range of values
- The general form of a floating-point number is:

$$(-1)^s m b^e$$

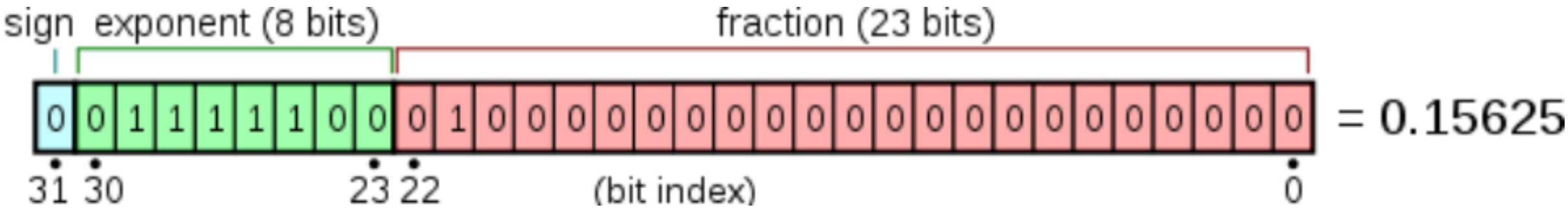
where,

  - $s$ is the sign bit
  - $m$ is the **significand**
  - $b$ is the **base**
  - $e$ is the **exponent**

- The base is a fixed value (normally 2)
- The significand and exponent take up a fixed number of bits

# IEEE 754

- The IEEE 754 is a standard for floating point numbers which most CPUs use
- Single precision numbers are 32 bits in length
- 1 bit is used for the sign
- 8 bits for the exponent
- 23 for the significand (an implicit leading bit is added for normalized numbers)



# IEEE 754

- There are three types of floating-point numbers:

  - **subnormal** numbers (the exponent is 0x00) which use the formula:

$$(-1)^s \times 0.m \times 2^{(-126)}$$

  - **normalized** numbers (the exponent is between 0x01 and 0xFE) which use the formula:

$$(-1)^s \times 1.m \times 2^{(e-127)}$$

  - **special** numbers (the exponent is 0xFF) if m=0 we have +- infinity, otherwise we have NaN.