CSE 351 Lecture 6 – Floating Point I

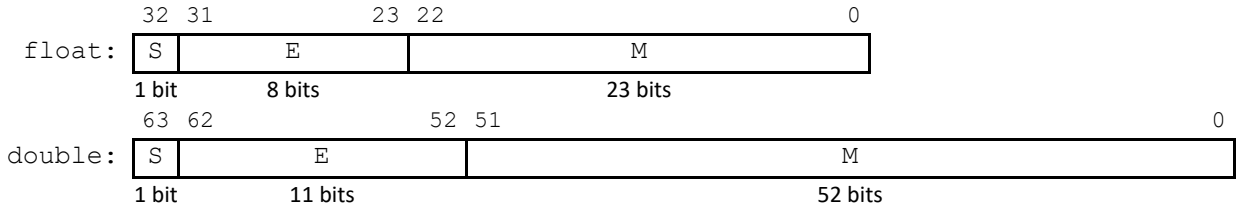
Scientific Notation

Scientific notation is a special way of writing numbers that explicitly shows a multiplication with a power of the base in the form of: numeral × base^{power}. The numeral can contain a "point" (*e.g.* decimal point) to allow us to express fractional numbers and the power can be used to "shift" the point to the left or to the right in the numeral (*e.g.* $3.51 \times 10^1 = 0.351 \times 10^2 = 35.1 \times 10^0$). Digits to the right of the point have weights of the base raised to *negative* exponents, starting on the left with base^{-1} and decreasing to the right. Scientific notation allows us to more easily express very small and very large numbers without having to write out so many digits.

The form and terminology we will use is: $\mathbf{sign} \times \mathbf{mantissa} \times \mathbf{base}^{\mathbf{exponent}}$, where the mantissa is always a positive number. A number written in scientific notation is considered ***normalized*** if there is *exactly* one non-zero digit to the left of point (*e.g.* 3.51×10^1).

IEEE 754 Floating Point Encoding

The accepted standard for floating point (FP) encoding is based on *normalized scientific <u>binary</u> notation* and encodes the three different parts (sign, mantissa, exponent) into separate fields (S, M, and E). Note that the single-letter field names (*e.g.* `M`) refer to the *binary encoding* of their corresponding values (*e.g.* the mantissa). There are two main different levels of precision (not drawn to scale):



The fields are translated separately according to: $\text{sign} \times \text{mantissa} \times 2^{\text{exponent}} = (-\mathbf{1})^{\mathbf{S}} \times \mathbf{1}.\mathbf{M} \times \mathbf{2}^{\mathbf{E-bias}}$

Sign Field (`S`)

`0` means positive and `1` means negative. We can mathematically write this relationship out as $(-1)^{S}$.

Exponent Field (`E`)

The exponent, which is always an integer value, is encoded in ***biased notation***: its value is shifted by the **bias of $2^{w-1} - 1$**, where *w* is the width of the exponent field, and then converted to unsigned. This allows us to represent about the same number of positive and negative exponents (it is useful to notice that the bias is represented exactly by a zero followed by all ones: `0b01…1`). The relationship can be written as **`E` = exponent value + bias**. Conversion examples in `float`, where bias = 2^{8-1}-1 = 127:

<u>Encode</u>: If value has exponent = 1 → encode 1 + 127 → store `E` = 0b 1000 0000

<u>Decode</u>: If encoding has `E` = 0b0100 0000 = 64 → value has exponent = 64 – 127 = -63

Mantissa Field (M)

The mantissa in a normalized scientific binary number must be of the form 1.xxx… because the only non-zero symbol we have is 1. Because of this, the relationship is **mantissa = 1.M** and we store the first bits *after* (to the right of) the binary point in our limited number of mantissa field bits. The leading 1 is *implicit* and not actually stored anywhere in the floating point encoding. If the mantissa (without any trailing 0's) is longer than the width of the mantissa field, then it is impossible to represent the exact value, leading to ***rounding errors***. Conversion examples in `float`, where the width of M is 23 bits:

Encode: A mantissa of 1.10111 is encoded as M = 0b 1011 1000 0000 0000 0000 000
Decode: M = 0b1101 0000 0000 0000 0000 000 is decoded as a mantissa of 1.1101

Normalized Floating Point Conversions

Translating to and from normalized floating point number follow these procedures:

FP → Decimal	**Decimal → FP**
1. Append the bits of M to implicit leading 1 to form the mantissa. 2. Multiply the mantissa by 2^{E-bias}. 3. Multiply the sign $(-1)^{S}$. 4. Multiply out the exponent by shifting the binary point. 5. Convert from binary to decimal.	1. Convert decimal to binary. 2. Convert binary to normalized scientific notation. 3. Encode sign as S (0/1). 4. Add the bias to exponent and encode E as unsigned. 5. The first bits after the leading 1 that fit are encoded into M.