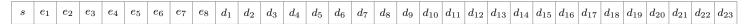
IEEE 754: What it means for your computer, and humanity

The textbook¹ has an idealized view of floating point, which I think is wise. But, in this separate document, I lay out the basics of how the *real* standard is *actually* implemented on a computer.

- Computer memories are organized into *bytes*, that is, groups of 8 *bits*, or collections of bytes called *words*. A bit is the irreducible atom of memory, which is in either of two states {0,1}. Though not related to the IEEE standard, note that *integers* are represented on computers using 1, 2, 4, or 8 bytes, thus 8, 16, 32, or 64 bit words, depending on the desired range.
- The IEEE 754 standard is about how *real* numbers are approximately represented in memory, that is, how *floating point* numbers are represented. "Floating point" is essentially just scientific notation, but using only finitely-many bits and thus representing only a finite subset of real numbers.
- The two best-known floating point representations use 32 ("single") and 64 ("double") bits. In single, the number

$$x = (-1)^s \times (1.d_1d_2d_3...d_{23})_2 \times 2^{(e_1...e_8)_2 - 127}$$

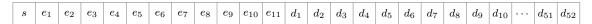
is represented by 32 bits this way:



In double, the number

$$x = (-1)^s \times (1.d_1d_2d_3...d_{52})_2 \times 2^{(e_1...e_{11})_2 - 1023}$$

is represented by 64 bits this way:



- Note that the "1." in the above single and double representations, which appears before the d_i bits, is always there. Thus it does not use a bit of memory! It is called the "implicit leading bit".
- The IEEE 754 standard is slightly more abstract than the concrete way the bits are arranged above. The standard says that every representable *nonzero* number is of the form

(1)
$$x = (-1)^s \times \frac{m}{\beta^{t-1}} \times \beta^e$$

for fixed positive integers β (the *base*) and t (the *precision*). The other numbers, namely $s \in \{0,1\}$ (the *sign*), the integer m (the *mantissa*), and the integer e (the *exponent*), depend on, and determine, x. These satisfy

(2)
$$\beta^{t-1} \le m \le \beta^t - 1,$$

$$e_{min} \le e \le e_{max}.$$

¹L. Trefethen and D. Bau, Numerical Linear Algebra, SIAM Press, 1997.

²"bit" = binary digit

- Unlike the system **F** in the textbook, in IEEE and other actual floating-point representations there are only finitely-many allowed values of the exponent *e*, and thus only finitely-many representable floating point numbers.
- In the current version of the standard, IEEE 754-2008, there are five basic formats, but two of these are (oddly enough) *decimal* standards with $\beta = 10$, and rarely used.
- The three basic formats that do matter the most are *binary*, that is, they have base $\beta = 2$. They use 32, 64, or 128 bits, respectively. We have already shown how the first two are implemented in memory in actuality. In terms of form (1) and constraints (2), they follow this table:

name	common name	precision t	exponent bits	exponent bias	e_{min}	e_{max}
binary32	single	24	8	$2^7 - 1 = 127$	-126	+127
binary64	double	53	11	$2^{10} - 1 = 1023$	-1022	+1023
binary128	quadruple	113	15	$2^{14} - 1 = 16383$	-16382	+16383

• If you convert the above binary description to decimal you get these heuristic values:

name	decimal precision	decimal e_{max}	decimal e_{min}
binary32	7.22	38.23	-37.93
binary64	15.95	307.95	-307.65
binary128	34.02	4931.77	-4931.47

• Regarding the exponent, if all bits e_i are zero or all are one then the number has special meaning. That is, for normal numbers in single the standard requires

$$(e_1 \dots e_8)_2 \in \{1, 2, \dots, 254\}$$

and in double the standard requires

$$(e_1 \dots e_{11})_2 \in \{1, 2, \dots, 2046\}.$$

- Representing the number zero, which is *not* in form (1), is an example of "special meaning." It is done by setting all bits other than s to zero. Oddly enough, this means "+0" and "-0" both exist on our computers as separate representations.
- Also there are representations of $+\infty$ and $-\infty$, of things that are "not a number" ("NaN"), and of things called "subnormal" numbers. For subnormal numbers, in the single representation $(e_1 \dots e_8)_2 = 0$ but not all bits d_i are zero.
- The IEEE 754 standard also addresses the rounding errors which occur with operations (addition, multiplication, etc.). In essence, the goal is that axiom (13.7) applies. But this is beyond my scope.
- Note "IEEE" stands for "Institute of Electrical and Electronics Engineers". For more on IEEE 754 see the wikipedia page