Floating-point Formats

Several different representations of real numbers have been proposed, but by far the most widely used is the floating-point representation.[1] Floating-point representations have a base β (which is always assumed to be even) and a precision p. If β = 10 and p = 3, then the number 0.1 is represented as 1.00×10^{-1}. If β = 2 and p = 24, then the decimal number 0.1 cannot be represented exactly, but is approximately $1.10011001100110011001101 \times 2^{-4}$.

In general, a floating-point number will be represented as $\pm$ d.dd... d $\times \beta^e$, where d.dd... d is called the significand[2] and has p digits. More precisely $\pm d_0 . d_1 d_2 \ldots d_{p-1} \times \beta^e$ represents the number

(1) $\pm(d_0 + d_1\beta^{-1} + \ldots + d_{p-1}\beta^{-(p-1)})\beta^e, (0 \le d_i < \beta)$.

The term floating-point number will be used to mean a real number that can be exactly represented in the format under discussion. Two other parameters associated with floating-point representations are the largest and smallest allowable exponents, e_{max} and e_{min}. Since there are β^p possible significands, and $e_{max} - e_{min} + 1$ possible exponents, a floating-point number can be encoded in

$$\lceil\log_2(e_{max} - e_{min} + 1)\rceil + \lceil\log_2(\beta^p)\rceil + 1$$

bits, where the final +1 is for the sign bit. The precise encoding is not important for now.

There are two reasons why a real number might not be exactly representable as a floating-point number. The most common situation is illustrated by the decimal number 0.1. Although it has a finite decimal representation, in binary it has an infinite repeating representation. Thus when β = 2, the number 0.1 lies strictly between two floating-point numbers and is exactly representable by neither of them. A less common situation is that a real number is out of range, that is, its absolute value is larger than $\beta \times \beta^{e_{max}}$ or smaller than $1.0 \times \beta^{e_{min}}$. Most of this paper discusses issues due to the first reason. However, numbers that are out of range will be discussed in the sections Infinity and Denormalized Numbers.

Floating-point representations are not necessarily unique. For example, both 0.01×10^1 and 1.00×10^{-1} represent 0.1. If the leading digit is nonzero ($d_0 \neq 0$ in equation (1) above), then the representation is said to be normalized. The floating-point number 1.00×10^{-1} is normalized, while 0.01×10^{1} is not. When $\beta = 2$, $p = 3$, $e_{min} = -1$ and $e_{max} = 2$ there are 16 normalized floating-point numbers, as shown in FIGURE D-1. The bold hash marks correspond to numbers whose significand is 1.00. Requiring that a floating-point representation be normalized makes the representation unique. Unfortunately, this restriction makes it impossible to represent zero! A natural way to represent 0 is with $1.0 \times \beta^{e_{min}-1}$, since this preserves the fact that the numerical ordering of nonnegative real numbers corresponds to the lexicographic ordering of their floating-point representations.[3] When the exponent is stored in a k bit field, that means that only $2^k - 1$ values are available for use as exponents, since one must be reserved to represent 0.

Note that the × in a floating-point number is part of the notation, and different from a floating-point multiply operation. The meaning of the × symbol should be clear from the context. For example, the expression $(2.5 \times 10^{-3}) \times (4.0 \times 10^{2})$ involves only a single floating-point multiplication.

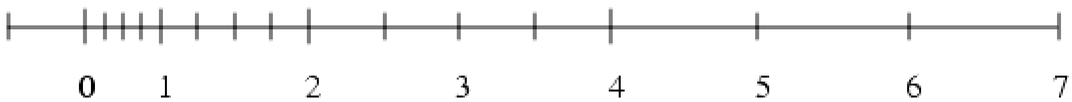

FIGURE D-1 Normalized numbers when $\beta = 2$, $p = 3$, $e_{min} = -1$, $e_{max} = 2$

Relative Error and Ulps