

B.2 IEEE FLOATING-POINT STANDARD 754

Until about 1980, each computer manufacturer had its own floating-point format. Needless to say, all were different. Worse yet, some of them actually did arithmetic incorrectly because floating-point arithmetic has some subtleties not obvious to the average hardware designer.

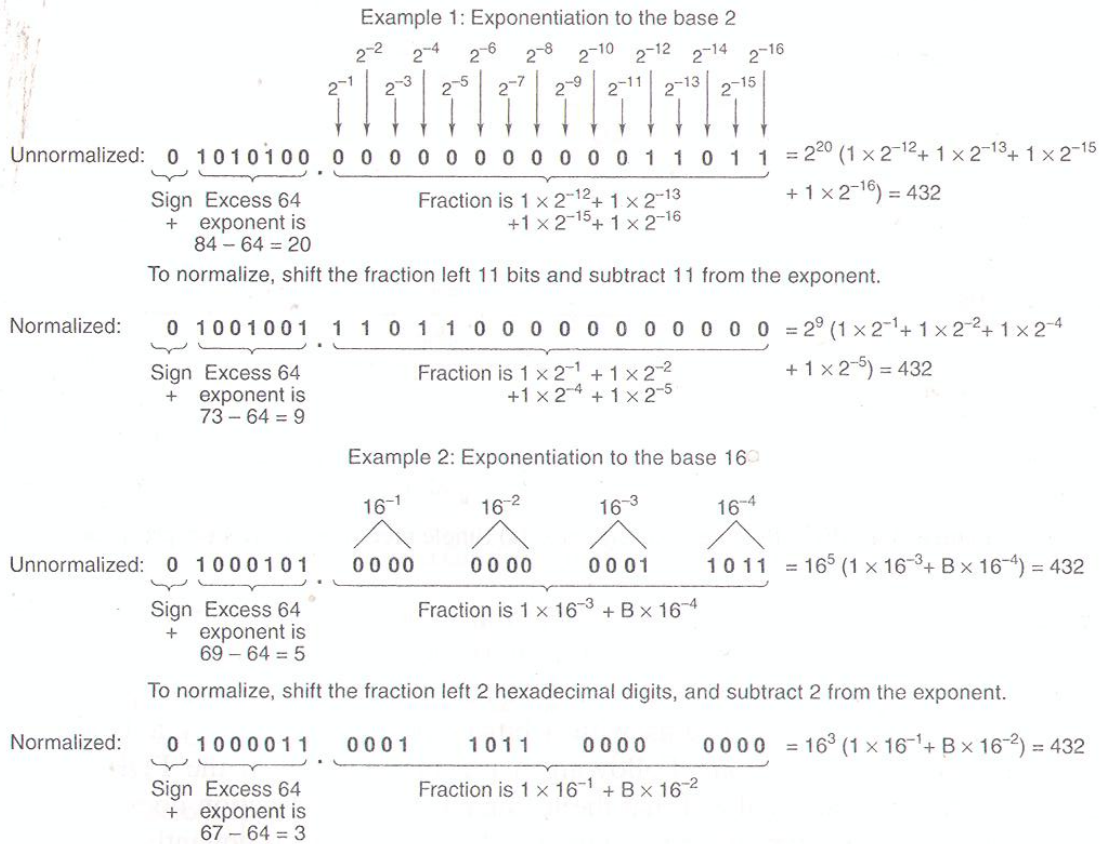


Figure B-3. Examples of normalized floating-point numbers.

To rectify this situation, in the late 1970s IEEE set up a committee to standardize floating-point arithmetic. The goal was not only to permit floating-point data to be exchanged among different computers but also to provide hardware designers with a model known to be correct. The resulting work led to IEEE Standard 754 (IEEE, 1985). Most CPUs these days (including the Intel, SPARC, and JVM ones studied in this book) have floating-point instructions that conform to the IEEE floating-point standard. Unlike many standards, which tend to be wishy-washy compromises that please no one, this one is not bad, in large part because it was primarily the work of one person, Berkeley math professor William Kahan. The standard will be described in the remainder of this section.

The standard defines three formats: single precision (32 bits), double precision (64 bits), and extended precision (80 bits). The extended-precision format is intended to reduce roundoff errors. It is used primarily inside floating-point arithmetic units, so we will not discuss it further. Both the single- and double-precision formats use radix 2 for fractions and excess notation for exponents. The formats are shown in Fig. B-4.

Both formats start with a sign bit for the number as a whole, 0 being positive and 1 being negative. Next comes the exponent, using excess 127 for single

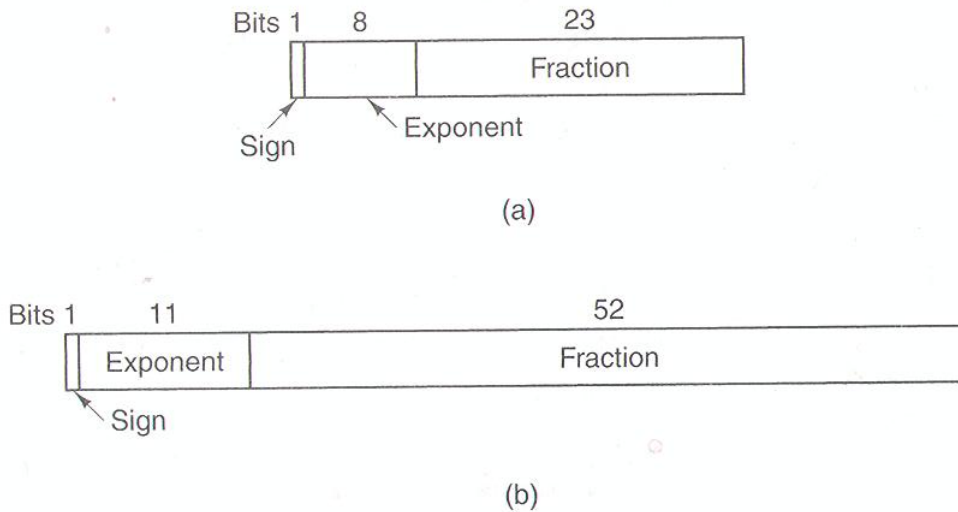


Figure B-4. IEEE floating-point formats. (a) Single precision. (b) Double precision.

precision and excess 1023 for double precision. The minimum (0) and maximum (255 and 2047) exponents are not used for normalized numbers; they have special uses described below. Finally, we have the fractions, 23 and 52 bits, respectively.

A normalized fraction begins with a binary point, followed by a 1 bit, and then the rest of the fraction. Following a practice started on the PDP-11, the authors of the standard realized that the leading 1 bit in the fraction does not have to be stored, since it can just be assumed to be present. Consequently, the standard defines the fraction in a slightly different way than usual. It consists of an implied 1 bit, an implied binary point, and then either 23 or 52 arbitrary bits. If all 23 or 52 fraction bits are 0s, the fraction has the numerical value 1.0; if all of them are 1s, the fraction is numerically slightly less than 2.0. To avoid confusion with a conventional fraction, the combination of the implied 1, the implied binary point, and the 23 or 52 explicit bits is called a **significand** instead of a fraction or mantissa. All normalized numbers have a significand, s , in the range $1 \leq s < 2$.