size. Note, however, that the gap between 0 and the smallest positive number is *much* larger than the gap between the smallest and next smallest positive number. This is the case with single- and double-precision floating-point numbers as well. The smallest positive (normalized) floating-point number in any such system is $1.0 \times 2^{-E}$, where $-E$ is the smallest representable exponent; the next smallest number is $(1+\epsilon) \times 2^{-E}$, where $\epsilon$ is the machine precision, so the gap between these two numbers is $\epsilon$ times the gap between 0 and the first positive number. This gap can be filled in using **subnormal** numbers. Subnormal numbers have less precision than normalized floating-point numbers and will be described in the next section.

## 5.4 IEEE FLOATING-POINT ARITHMETIC

In the 1960s and 1970s, each computer manufacturer developed its own floating-point system, resulting in inconsistent program behavior across machines. Most computers used binary arithmetic, but the IBM 360/70 series used hexadecimal (base 16), and Hewlett-Packard calculators used decimal arithmetic. In the early 1980s, due largely to the efforts of W. Kahan, computer manufacturers adopted a standard: the IEEE (Institute of Electrical and Electronics Engineers) standard. This standard required:

- Consistent representation of floating-point numbers across machines.
- Correctly rounded arithmetic.
- Consistent and sensible treatment of exceptional situations such as divide by 0.

In order to comply with the IEEE standard, computers represent numbers in the way described in the previous section. A special representation is needed for 0 (since it cannot be represented with the standard hidden bit format), and also for $\pm\infty$ (the result of dividing a nonzero number by 0), and also for NaN (Not a Number; e.g., 0/0). This is done with special bits in the exponent field, which slightly reduces the range of possible exponents. Special bits in the exponent field are also used to signal subnormal numbers.

There are 3 standard precisions. As noted previously, a **single-precision** word consists of 32 bits, with 1 bit for the sign, 8 for the exponent, and 23 for the significand. A **double-precision** word consists of 64 bits, with 1 bit for the sign, 11 for the exponent, and 52 for the significand. An **extended-precision** word consists of 80 bits, with 1 bit for the sign, 15 for the exponent, and 64 for the significand. [Note, however, that numbers stored in extended precision do not use hidden-bit storage.]

Table 5.2 shows the floating-point numbers, subnormal numbers, and exceptional situations that can be represented using IEEE double precision.

It can be seen from the table that the smallest positive normalized floating-point number that can be stored is $1.0_2 \times 2^{-1022} \approx 2.2 \times 10^{-308}$, while the largest is $1.1\ldots1_2 \times 2^{1023} \approx 1.8 \times 10^{308}$. The exponent field for normalized

TABLE 5.2
IEEE double precision.

| If exponent field is | Then number is | Type of number: |
|---|---|---|
| 00000000000 | $\pm(0.b_1\ldots b_{52})_2 \times 2^{-1022}$ | 0 or subnormal |
| $00000000001 = 1_{10}$ | $\pm(1.b_1\ldots b_{52})_2 \times 2^{-1022}$ | Normalized number |
| $00000000010 = 2_{10}$ | $\pm(1.b_1\ldots b_{52})_2 \times 2^{-1021}$ | |
| $\vdots$ | $\vdots$ | Exponent field is |
| $01111111111 = 1023_{10}$ | $\pm(1.b_1\ldots b_{52})_2 \times 2^{0}$ | (actual exponent) + 1023 |
| $\vdots$ | $\vdots$ | |
| $11111111110 = 2046_{10}$ | $\pm(1.b_1\ldots b_{52})_2 \times 2^{1023}$ | |
| 11111111111 | $\pm\infty$ if $b_1 = \ldots = b_{52} = 0$, NaN otherwise | Exception |

The two special exponent field bit patterns are all 0s and all 1s. An exponent field consisting of all 0s signals either 0 or a subnormal number. Note that subnormal numbers have a 0 in front of the binary point instead of a 1 and are always multiplied by $2^{-1022}$. Thus the number $1.1_2 \times 2^{-1024} = 0.011_2 \times 2^{-1022}$ would be represented with an exponent field string of all 0s and a significand field 0110...0. Subnormal numbers have less precision than normalized floating-point numbers since the significand is shifted right, causing fewer of its bits to be stored. The smallest positive subnormal number that can be represented has fifty-one 0s followed by a 1 in its significand field, and its value is $2^{-52} \times 2^{-1022} = 2^{-1074}$. The number 0 is represented by an exponent field consisting of all 0s and a significand field of all 0s. An exponent field consisting of all 1s signals an exception. If all bits in the significand are 0, then it is $\pm\infty$. Otherwise it represents NaN.

### WILLIAM KAHAN

William Kahan is an eminent mathematician, numerical analyst and computer scientist who has made important contributions to the study of accurate and efficient methods of solving numerical problems on a computer with finite precision. Among his many contributions, Kahan was the primary architect behind the IEEE 754 standard for floating-point computation. Kahan has received many recognitions for his work, including the Turing Award in 1989 and being named an ACM Fellow in 1994. Kahan was a professor of mathematics and computer science and electrical engineering at the University of California, Berkeley, and he continues his contributions to the