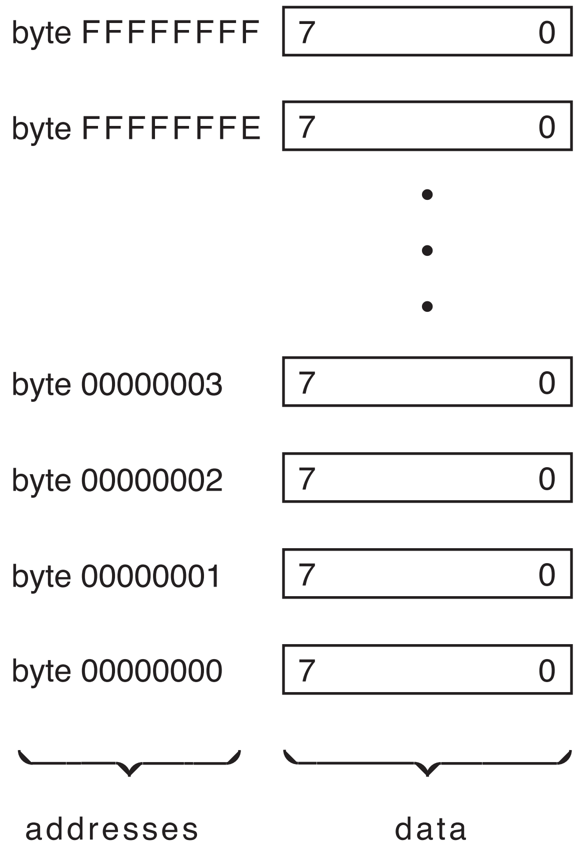

Figure 1.1. Use of 32-bit unsigned integers to address bytes in RAM.

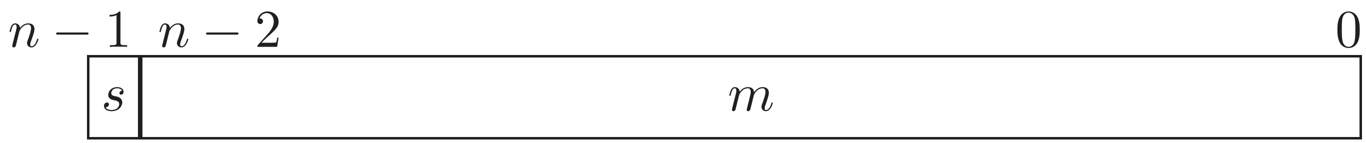

Figure 1.2. The bit sequence of the *n*-bit, binary, unsigned-integer representation.

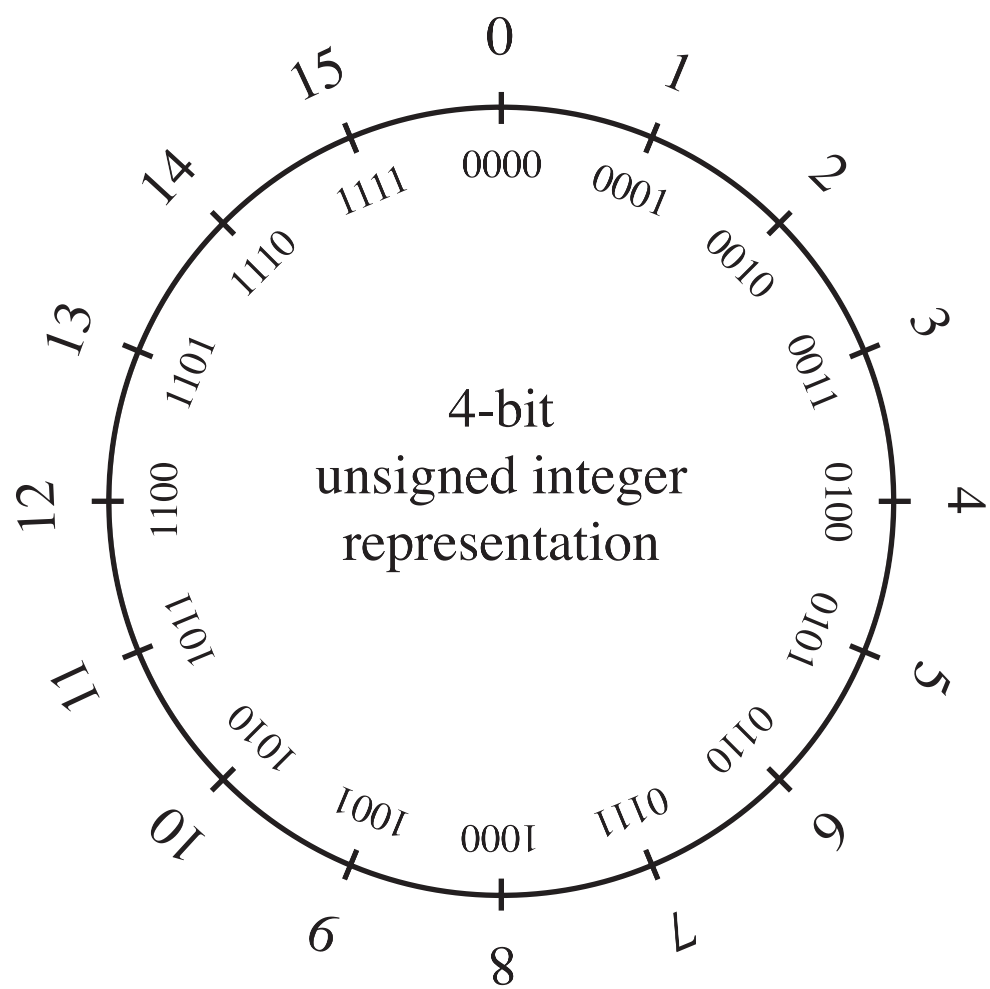

Figure 1.3. The 4-bit, binary, unsigned-integer representation.

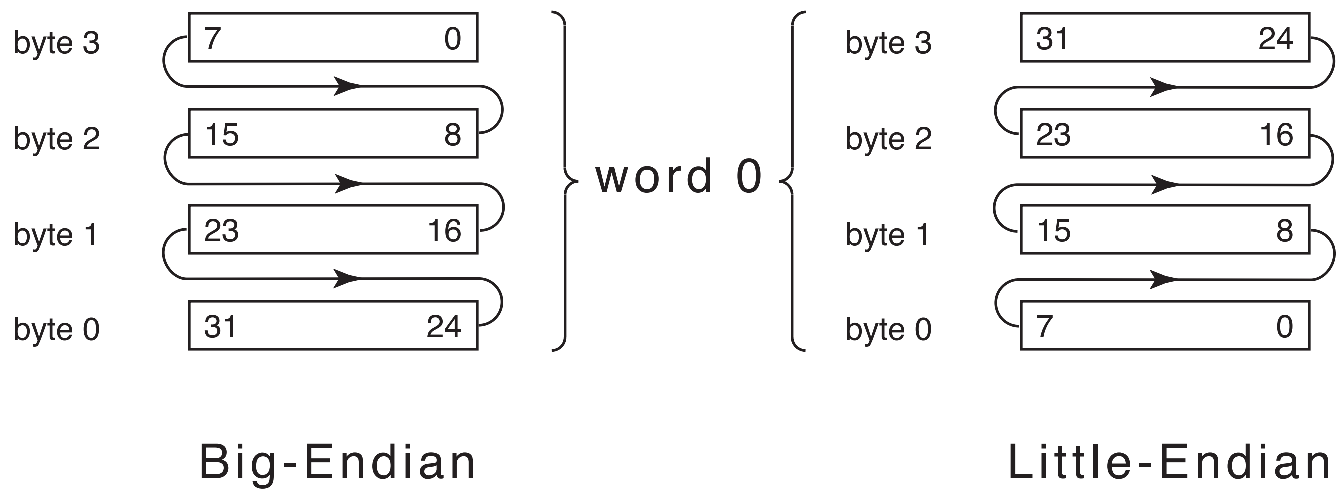

Figure 1.4. Big-endian and little-endian byte orderings. For clarity, only one word of memory is shown. The address of the word is the address of the byte with the lowest address within the word. The arrows point in the direction of increasingly significant digits.

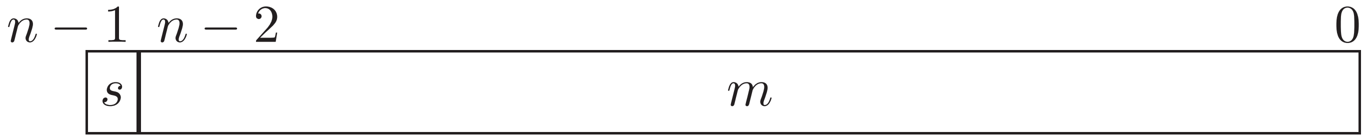

Figure 1.5. The bit sequence of an *n*-bit, binary, sign-magnitude representation.

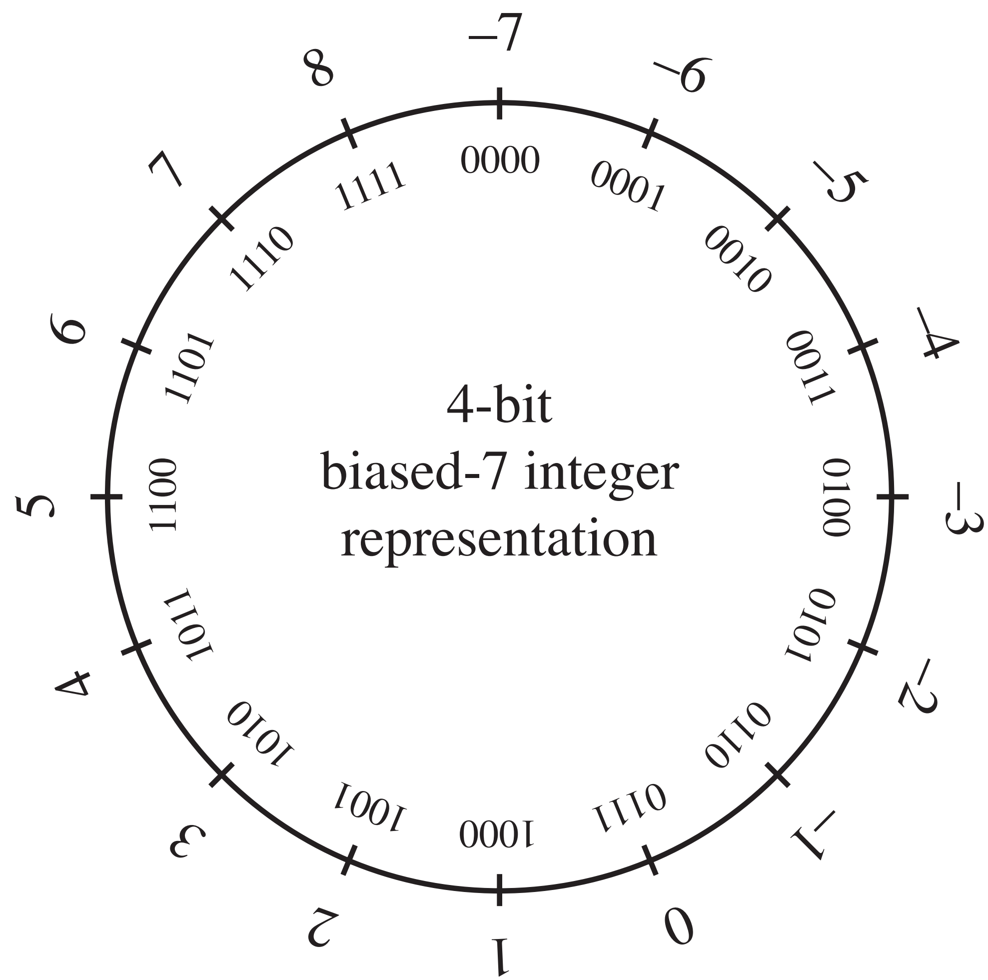

Figure 1.6. The 4-bit, binary, biased-7 integer representation.

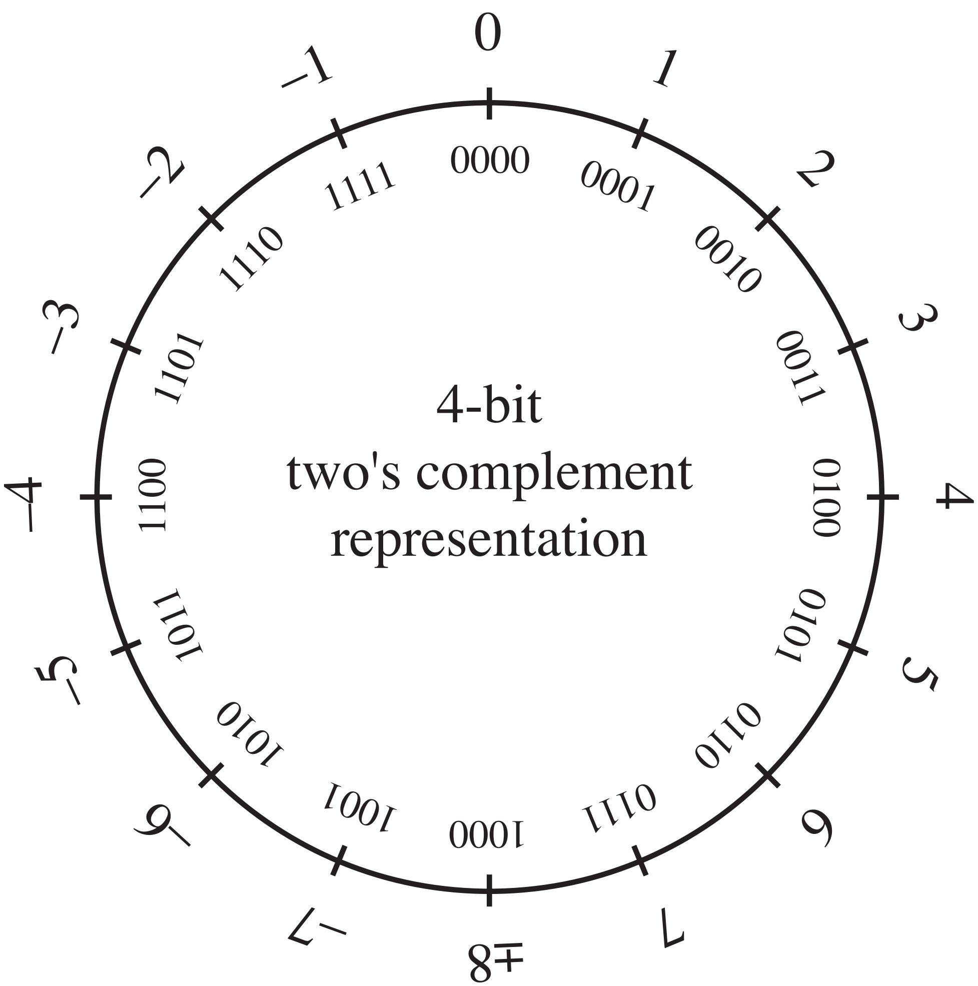

Figure 1.7. The 4-bit, two's complement integer representation.

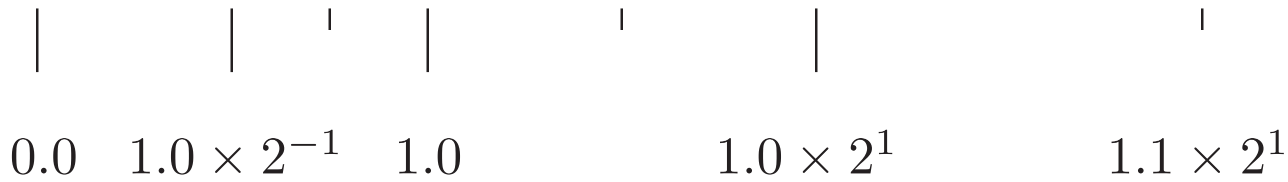

Figure 1.8. This figure shows all of the nonnegative base-2 floating-point numbers in a toy floating-point representation with two significant digits, a maximum exponent of 1, and a minimum exponent of –1. The negative, nonzero floating-point numbers that exist in this representation are not shown.

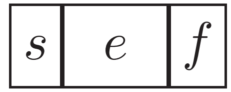

Figure 1.9. A bit ordering that implements the toy floating-point representation. The significand of a floating-point number represented in this format is equal to $1.f$. The initial 1, which is not stored in order to save space, is called an **implicit 1 bit**. Note that the exponent is biased.

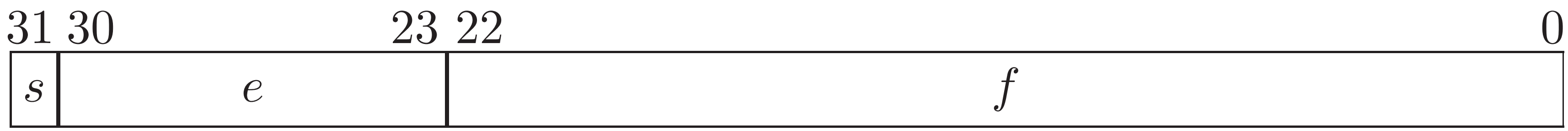

Figure 1.10. The bit sequence of the IEEE-754 single-precision representation.

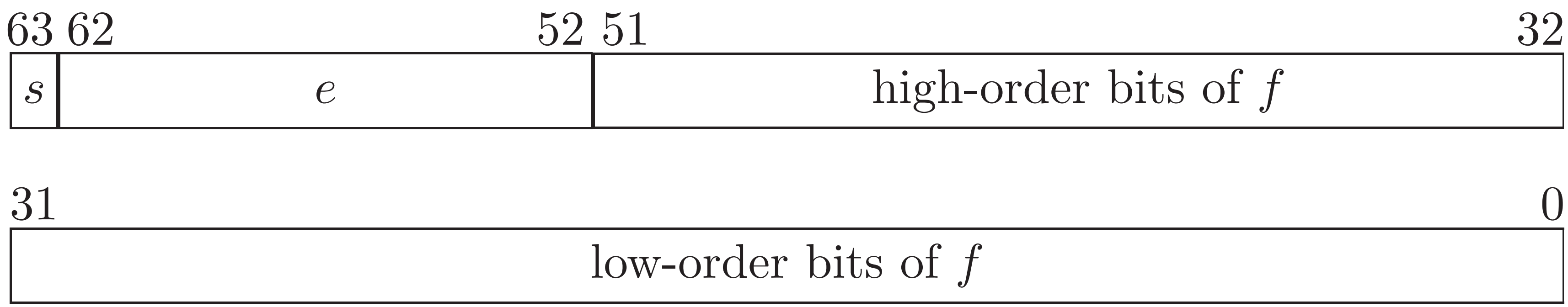

Figure 1.11. The bit sequence of the IEEE-754 double-precision representation.

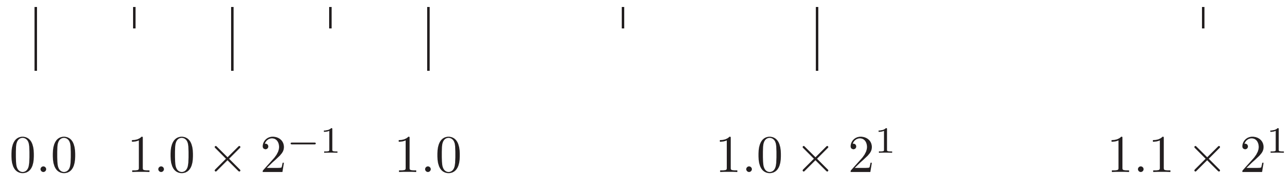

Figure 1.12. This figure shows an extra (denormalized) floating-point number between $r_{\min} = 1.0 \times 2^{-1}$ and 0 in the toy floating-point representation.