Instead, the point is to have a basic understanding of how computers store information and — most important — to understand how the hexadecimal counting system works (which I describe in the following section).

Here are some of the more interesting characteristics of binary and how the system is similar to and differs from the decimal system:

- The number of bits allotted for a binary number determines how large that number can be. If you allot eight bits, the largest value that number can store is 11111111, which happens to be 255 in decimal.
- TIP To quickly determine how many different values you can store in a binary number of a given length, use the number of bits as an exponent of two. An eight-bit binary number, for example, can hold 2^8 values. Because 2^8 is 256, an 8-bit number can have any of 256 different values, which is why a byte, which is eight bits, can have 256 different values.
- This powers-of-two concept is why computers don't use nice, even, round numbers in measuring such values as memory or disk space. A value of 1K, for example, isn't an even 1,000 bytes — it's 1,024 bytes because 1,024 is 2^{10}. Similarly, 1MB isn't an even 1,000,000 bytes but rather is 1,048,576 bytes, which happens to be 2^{20}.

Doing the logic thing

One of the great things about binary is that it's very efficient at handling special operations called *logical operations.* Four basic logical operations exist, although additional operations are derived from the basic four operations. Three of the operations — AND, OR, and XOR — compare two binary digits (bits). The fourth (NOT) works on just a single bit.

The following list summarizes the basic logical operations:

- AND: An AND operation compares two binary values. If both values are 1, the result of the AND operation is 1. If one value is 0 or both of the values are 0, the result is 0.
- OR: An OR operation compares two binary values. If at least one of the values is 1, the result of the OR operation is 1. If both values are 0, the result is 0.
- XOR: An XOR operation compares two binary values. If exactly one of them is 1, the result is 1. If both values are 0 or if both values are 1, the result is 0.