BINARY VS. DECIMAL

The binary number system may seem strange to you because you count using the ***decimal number system***, which uses the digits 0 through 9. Counting in the decimal number system comes very naturally to you because you learned it from a very young age. But someone invented the decimal number system just like someone invented the binary number system. The decimal number system is based on tens because you have ten fingers on your hands. The binary number system is based on twos because of the circuits in a computer. *Both systems, however, can be used to represent the same values.*

In the decimal number system, each digit of a number represents a power of 10. That is why the decimal number system is also called the base 10 number system. Consider the number 3208 for example. When you read that number, you automatically understand it to mean three thousands, two hundreds, no tens, and eight ones. Represented mathematically, you could say $(3 \times 1000) + (2 \times 100) + (0 \times 10) + (8 \times 1) = 3208$, as shown in Figure 2-4.

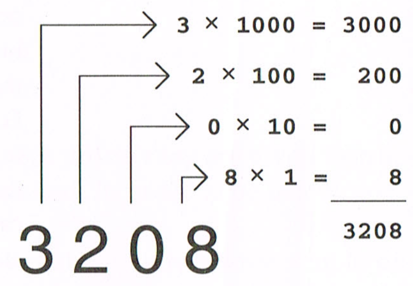

FIGURE 2-4
Each digit of the decimal number 3208 represents a power of 10.

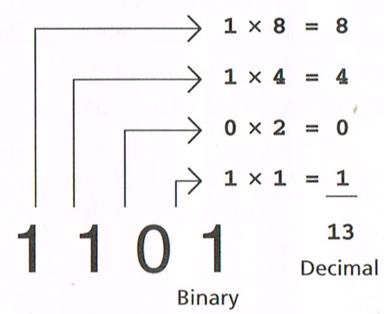

FIGURE 2-5
Each digit of the binary number 1101 represents a power of 2, so conversion to the decimal system is easy.

In the binary number system, each digit represents a power of 2, like you saw in Table 2-1. Working with powers of 2 is not as natural to you as working with powers of 10. But with a little practice you will see that base 2 numbers are not so mysterious. Consider the binary number 1101. Even though the number is four digits long, its value is nowhere near a thousand. The powers of 2 are 1, 2, 4, 8, 16, 32, and so on. So for this number, its decimal equivalent is $(1 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1) = 13$, as shown in Figure 2-5. So the binary number 1101 is equivalent to thirteen in the decimal number system.

Extra for Experts

Decimal Points and Binary Points

You have used decimal points for a long time. Did you know there is a binary point? A decimal point divides the ones place and the tenths place, or 10^0 from 10^{-1}. There is an equivalent in the binary number system called the binary point. It divides the 2^0 place from the 2^{-1} place.

With a binary point, it is possible to have binary numbers like 100.1, which in decimal is 4.5. Can you convert the binary number 10.01 to decimal? If you got 2.25 as the answer, you are correct. Try converting the binary number 11.001001 to decimal.

NUMBERS REPRESENTING LETTERS AND SYMBOLS

If all data in a computer is represented by numbers, how are letters and symbols stored? Letters and symbols, called ***characters***, are assigned a number that the computer uses to represent them. Most computers assign numbers to characters according to the ***American Standard Code for Information Interchange (ASCII)***. Figure 2-6 shows some of the ASCII (pronounced *ask-e*) codes. For a complete ASCII table, see Appendix A.

ASCII CODES

Character	Decimal Number System	Binary Number System
$	36	010 0100
*	42	010 1010
A	65	100 0001
B	66	100 0010
C	67	100 0011
D	68	100 0100
a	97	110 0001
b	98	110 0010
c	99	110 0011
d	100	110 0100

FIGURE 2-6
In the computer, each character is stored as a number.

The basic ASCII code is based on 7 bits, which gives 128 characters. About 95 of these are upper and lowercase letters, numbers, and symbols. Some of the characters are used as codes for controlling communication hardware and other devices. Others are invisible characters like Tab and Return. Most computers extend the ASCII code to 8 bits (a whole byte) to represent 256 characters. The additional 128 characters are used for graphical characters and characters used with foreign languages.

REPRESENTING INSTRUCTIONS

You have seen how computers use bits to represent data. We use computers to do much more than represent data for storage. Computers are useful because they follow instructions to do work. And just like data, instructions are represented by combinations of bits.

Recall that the microprocessor is the device in which instructions are executed. Each instruction consists of ones and zeros, called ***machine language***.

Writing a program in machine language would be very difficult because even a simple program requires hundreds or even thousands of microprocessor instructions. Another problem is that the numbers used to represent microprocessor instructions are difficult for people to understand. Figure 2-7 shows a short machine language program. Each line is one instruction for the microprocessor. To find out what the program does, you would have to look up the machine language instructions in the microprocessor's reference manual.